DOCUMENT RESUME

ED 187 570	SE 030 972
AUTHOR	Warfield, John N.
TITLE	Development of an Interpretive Structural Model and Strategies for Implementation Based on a Descriptive and Prescriptive Analysis of Resources for Environmental Education/Studies. A Sourcebook for the Design of a Regional Environmental Learning System, Volume IV: Conducting Collective Inquiry.
INSTITUTION	Virginia Univ., Charlottesville. School of Engineering and Applied Science.
SPONS AGENCY	Office of Education (DHEW), Washington, D.C. Office of Environmental Education.
REPORT NO	UVA/522032/EE79/124
PUB DATE	31 Aug 79
CONTRACT	300-700-4028
NOTE	381p.: For related documents, see SE 030 969-974 and ED 173 172.
EDRS PRICE	MF01/PC16 Plus Postage.
DESCRIPTORS	*Change Strategies; Curriculum Development; *Decision Making; *Educational Assessment; Educational Needs; *Educational Planning; Educational Research; Elementary Secondary Education; *Environmental Education; Models; Nonformal Education; Postsecondary Education; Science Education; *Simulation

ABSTRACT

The steps, approaches, and tools of the collective inquiry process posited by a Regional Environmental Learning System is the subject of this volume. Approaches discussed include: (1) charette approach, (2) AT&T/Battelle approach, and (3) the Washington State approach. Tools for collective inquiry are described and field tests are discussed. References and four topical bibliographies are provided. An appendix presents a computer program for modeling technique proposed for use in collective inquiry. (RE)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED187570

U.S. DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
NATIONAL INSTITUTE OF
EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY.

RESEARCH LABORATORIES FOR THE ENGINEERING SCIENCES

SCHOOL OF ENGINEERING AND APPLIED SCIENCE

UNIVERSITY OF VIRGINIA

Charlottesville, Virginia 22901

DEVELOPMENT OF AN INTERPRETIVE STRUCTURAL MODEL AND STRATEGIES FOR IMPLEMENTATION BASED ON DESCRIPTIVE AND PRESCRIPTIVE ANALYSIS OF RESOURCES FOR ENVIRONMENTAL EDUCATION/STUDIES

VOLUME IV

CONDUCTING COLLECTIVE INQUIRY

Submitted to:

Office of Environmental Educaton
Department of Health, Education and Welfare
400 Maryland Avenue, S.W.
FOB #6, Room 2025
Washington, D. C. 20202

Submitted by:

John N. Warfield

Report No. UVA/522032/EE79/124

August 1979

SE 030972

RESEARCH LABORATORIES FOR THE ENGINEERING SCIENCES

Members of the faculty who teach at the undergraduate and graduate levels and a number of professional engineers and scientists whose primary activity is research generate and conduct the investigations that make up the school's research program. The School of Engineering and Applied Science of the University of Virginia believes that research goes hand in hand with teaching. Early in the development of its graduate training program, the School recognized that men and women engaged in research should be as free as possible of the administrative duties involved in sponsored research. In 1959, therefore, the Research Laboratories for the Engineering Sciences (RLES) was established and assigned the administrative responsibility for such research within the School.

The director of RLES—himself a faculty member and researcher—maintains familiarity with the support requirements of the research under way. He is aided by an Academic Advisory Committee made up of a faculty representative from each academic department of the School. This Committee serves to inform RLES of the needs and perspectives of the research program.

In addition to administrative support, RLES is charged with providing certain technical assistance. Because it is not practical for each department to become self-sufficient in all phases of the supporting technology essential to present-day research, RLES makes services available through the following support groups: Machine Shop, Instrumentation, Facilities Services, Publications (including photographic facilities), and Computer Terminal Maintenance.

DEVELOPMENT OF AN INTERPRETIVE STRUCTURAL MODEL AND STRATEGIES FOR IMPLEMENTATION BASED ON A DESCRIPTIVE AND PRESCRIPTIVE ANALYSIS OF RESOURCES FOR ENVIRONMENTAL EDUCATION/STUDIES

A SOURCEBOOK FOR THE DESIGN OF A REGIONAL ENVIRONMENTAL LEARNING SYSTEM

VOLUME IV

CONDUCTING COLLECTIVE INQUIRY

Contract No. 300-700-4028

Work Supported Under the
Environmental Education Act of 1970
P. L. No. 91-516,
P. L. No. 93-278 and P. L. No. 95-482, as amended

Submitted to:

Office of Environmental Education
Department of Health, Education and Welfare
400 Maryland Avenue, S.W.
FOB #6, Room 2025
Washington, D. C. 20202

Submitted by:

John N. Warfield

Department of Electrical Engineering
RESEARCH LABORATORIES FOR THE ENGINEERING SCIENCES
SCHOOL OF ENGINEERING AND APPLIED SCIENCE
UNIVERSITY OF VIRGINIA
CHARLOTTESVILLE, VIRGINIA

Report No. UVA/522032/EE79/124 Copy No.________
August 31, 1979

A SOURCEBOOK FOR THE DESIGN
OF A
REGIONAL ENVIRONMENTAL LEARNING SYSTEM

VOLUME IV: CONDUCTING COLLECTIVE INQUIRY

PREFACE

This is one of six Volumes of a report which, collectively, is intended to be a Sourcebook for the Design of a Regional Environmental Learning System. The report was prepared under Contract 300-700-4028 with the Office of Environmental Education.

This six-volume report presumes some background concerning the concept of a Regional Environmental Learning System, and with environmental education as a whole. Considerable relevant background was supplied in Volume 9 of the 4th Quarterly Report (A Descriptive Analysis of Environmental Education) and in the 5th Quarterly Report (Conceptual Basis for the Design of Regional Environmental Learning Systems), both of which are available from the Office of Environmental Education.

Volume 1 contains an Overview of the Sourcebook, with short summaries of the other Volumes.

VOLUME IV

CONDUCTING COLLECTIVE INQUIRY

by

Robert W. House
Vanderbilt University

with contributions from

Andrew P. Sage
University of Virginia

and an appendix
COMPUTER IMPLEMENTATION OF ISM

by

Raymond L. Fitz
David R. Yingling
Joanne B. Troha
Karen O. Crim

University of Dayton

August 1979

TABLE OF CONTENTS

PREFACE ii

EXECUTIVE SUMMARY 1

COLLECTIVE INQUIRY STEPS 2

APPROACHES TO COLLECTIVE INQUIRIES 8

  1. CHARETTE APPROACH 9

  2. AT&T/BATTELLE APPROACH 13

  3. WASHINGTON STATE APPROACH 17

TOOLS FOR COLLECTIVE INQUIRY 20

  DESCRIPTIONS OF TOOLS 20

  FIELD TEST OF TOOLS 32

REFERENCES 38

COLLECTIVE INQUIRY BIBLIOGRAPHY 39

BIBLIOGRAPHY ON APPLICATIONS OF INTERPRETIVE STRUCTURAL MODELING 42

BIBLIOGRAPHY ON THE THEORY OF INTERPRETIVE STRUCTURAL MODELING 46

BIBLIOGRAPHY ON WORTH ASSESSMENT 48

APPENDIX: COMPUTER IMPLEMENTATION OF ISM 55

VOLUME IV

CONDUCTING COLLECTIVE INQUIRY

EXECUTIVE SUMMARY

As indicated in previous volumes, especially Volumes I and III, learning about one's environment encompasses a large number of factors, factors like resources, conservation, pollution, economics, demography, urban and regional planning, technology, and transportation. Understanding all these factors, their interrelations, and their implications for planning and managing one's future cannot be done alone. Involvement in collective inquiries with other participants with a variety of skills, knowledge, experiences, perceptions, and values is required. For such inquiries to be efficient and productive, means for managing the generation, structuring, communication, and documentation of ideas are needed, in other words, tools for idea management are needed. Through the use of such tools, the process of collective inquiry may enable individuals and groups to understand and improve their environments.

In addition to helping individuals, the process of collective inquiry is useful to teachers and others concerned with managing or facilitating learning. Collective inquiries can be exceedingly useful for achieving environmental education "that synthesizes and integrates pertinent subject matter across and between a variety of disciplines". In addition to idea management tools, approaches to planning and managing the conduct of collective inquiries are needed.

This volume provides some tools and approaches for conducting collective inquiries. Although an attempt is made to provide the reader with enough information to carry out an inquiry, the references provide more information and may be referred to if the information provided in this volume needs to be augmented.

As used here, "approaches to collective inquiries" mean suggestions and ideas for planning and managing the process of an inquiry. "Tools" mean particular ways to facilitate a particular step in the process.

COLLECTIVE INQUIRY STEPS

Why might one want to conduct a collective inquiry? The answer depends on the stage at which a collection of participants is dealing with an environmental issue. For different stages, different purposes are served. Some examples are:

to raise awareness of environmental issues and problems or opportunities,
to define or explain problems and issues concerning the environment,
to share perceptions or facts,
to discover values, beliefs, or assumptions,
to set goals,
to develop candidate policies, plans, or programs to achieve goals,
to assess the worth of candidate policies, plans, or programs,
to choose policies, plans, or programs,
to develop approaches for implementation of choices,
to identify who have what tasks to accomplish for implementation,
to evaluate performance of policies, plans, or programs.

To achieve any of these purposes, it is helpful to give careful consideration to the following steps for conducting an inquiry.

1. At the outset it is important to define clearly the purpose(s) of the collective inquiry. This is needed to insure that the participants work together toward a common goal. If it is not done, some participants may try to engage in activities that are incompatible with the activities of other participants during an inquiry. As an example, some may try to develop candidate policies addressed to an environmental issue while other participants at the same time are trying to assess the worth of policies. The former group is engaged in a process requiring conceptualization, while the latter is engaged in analysis. Mixing these kinds of activities at the same time often leads to conflicts and/or failure. An approach more likely to succeed would separate the activities in time by first generating candidate policies and then assessing the worth of each candidate. Defining the purpose clearly will help the group of participants to focus on a single task at any given point in time and, thereby, improve their chances for success. [1] [2]

Another reason for defining the purpose clearly is to help insure that appropriate participants, appropriate arrangements, and appropriate tools are obtained and involved in the inquiry. More is said on these below.

2. It is important to plan well for obtaining good participants and arrangements for an inquiry. Too often collective inquiries fail because not enough attention is given to one or both of these. Table I indicates some of the main points requiring attention.

TABLE I

PLANNING COLLECTIVE INQUIRIES

- GET GOOD PARTICIPANTS
- INSURE GOOD PHYSICAL ARRANGEMENTS
- INSURE GOOD STAFFING
- PREPARE SOURCE MATERIAL FOR THE INQUIRY
- INSURE GOOD RECEPTION AND ORIENTATION FOR PARTICIPANTS
- INSURE CONTRIBUTIONS ARE RECORDED
- PREVENT DISRUPTIONS TO ORDERLY THINKING
- SUMMARIZE AND FEED BACK RESULTS
- DOCUMENT AND ANALYZE INQUIRY PROCESS AND RESULTS
- DISTRIBUTE FINAL REPORT

Good participants have the right qualifications, are willing to contribute, and have an interest in the issue under consideration.

Good physical arrangements have adequate blackboards, flip charts, soundproofing, temperature control, audio-visual equipment, sleeping accommodations, and everything else needed to facilitate the productivity of the effort.

Good staffing insures adequate secretarial help, receptionists, etc., as well as resource persons with knowledge of the content of the issue in the inquiry.

In addition to resource persons, the content of the inquiry may be aided by providing "issue" or "background" papers for the participants to be read before the meeting and/or referred to at the meeting. Relevant "facts books" may be prepared also for use at the meeting.

A good reception for the participants reinforces in their minds that their contributions are important for the success of the meeting. Good orientation focuses their thinking on the purposes for the meeting, the processes to be employed to achieve those purposes, and the schedule to be followed. Good orientation also helps insure that no participants' contributions are lost because the participants are lost.

Recording the contributions of the participants insures that after the meeting one can refer to the record for whatever reason.

Preventing disruptions of the participants' work is important whether from telephone calls, the need for minor decisions, lobbyists for special interests, or whatever.

Summarizing and feeding back results is useful during and after the meeting. Errors can be corrected. Results can be improved through iteration. The amount of progress can be assessed.

Documenting the process and the results of the inquiry serves several purposes. It allows analysis both by the participants and by others not involved. It provides a springboard for later stages of inquiry. And it provides valuable information for improving the tools for collective inquiry. It allows others not involved in the process to share in the learning achieved.

None of this can be achieved without careful and persistent attention to detailed planning.

3. After the purpose for the inquiry is defined and the participants and arrangements are fixed, attention can turn to the inquiry process as such. In

most cases, the participants first need to identify the elements of the issue being addressed. For example, if the purpose of the collective inquiry is to define an environmental problem, different participants can probably see different contributing factors. These factors put together make up the environmental problem. Each is an element of the problem; none is the whole problem. Through the process of generating the elements, the participants gain an improved and shared understanding of the whole problem.

4. After the elements of the issue under consideration have been generated, it is natural in most circumstances to begin to consider what the relations are among the elements. Transitive relations are useful for developing chains of relations. For example, suppose the elements of the issue are the contributing factors of an environmental problem. Consider the relation "contributes to". Then if factor A contributes to factor B, and if factor B contributes to factor C, then, by transitivity, factor A contributes to factor C.

Choosing the relation to use is an important decision in most inquiries. Consider for a moment the differences in the following: "contributes to"; "is more important than"; "is more easily managed than"; and "occurs before". Depending on which one of these is chosen, in most cases, different structures of related elements will result. It is sometimes useful to consider more than one relation with a set of elements, but not at the same time.

5. After a contextual relation is chosen, the participants may begin to consider pairs of elements to decide whether the relation holds between them. In the process of deciding whether the relation holds, two situations can arise. The first occurs when the group decides the relation holds in both directions. For example, the group might decide the chicken is more important than the egg and that the egg is more important than the chicken, in which case, it seems natural to agree they are of equal importance. Such situations can occur with

many relations, e.g. "contributes to". The second situation occurs when the group finds it impossible to decide that the relation holds in either direction, e.g. A is not more important than B, and B is not more important than A. There is no requirement that the relation hold in either direction, and no one should be concerned if it does not.

In the process of deciding whether the relation holds between two elements, the group often develops an improved definition or understanding of the elements or the relation. They also often gain a better understanding of other participants' views about the elements or the values, beliefs, or perceptions of other participants. These improved understandings are among the main, beneficial outcomes of collective inquiries.

This step ends when all the pairs of elements have been tested with the chosen relation either directly, or by inference using transitivity. By using the latter, the group usually does not have to decide a large number of cases. That is, if A relates to B and B relates to C, then it can be inferred immediately that A relates to C without asking the group to decide. The use of a computer greatly facilitates making such inferences.

6. Once the relation has been tested with all pairs of elements and the results have been displayed, then the group can analyze the results to determine if they seem intuitively reasonable or, failing that, determine an explanation for the resulting structure of related elements. If this is possible, new insights about the issue under consideration are often gained.

If the results do not seem reasonable and no explanation can be found, then the group may choose to amend the structure to achieve one that does seem intuitively reasonable or that can be explained. One of the ways to amend the structure is to reconsider decisions where the group was almost evenly divided on whether the relation held or not. Through additional discussion, the group may decide to reverse the decision. This will alter the structure. Another way to amend the structure is to recognize and decide that a major portion of the structure belongs in a different place within the overall structure. In these cases and others the amendment process is facilitated by the use of a computer.

Several different structures may be achieved by using different elements, or different relations, or different groups of participants. In these cases and in others, there may be a need to rank or assess the worth of the various structures.

7. When the collective inquiry is completed, documentation of the results and the process is useful for several reasons. It allows others who were not participants to gain some understanding of how the results were achieved. If they see weak points, these can be strengthened. Another benefit is to enable other participants to build on the results. A third use is for reference by the participants or others who may want to look back at the process or the results.

After documentation, the results should be disseminated to all of those who need to be aware of or to use the results for follow-up actions.

APPROACHES TO COLLECTIVE INQUIRIES

Approaches to collective inquiries considers the questions of who, where, what, and when. Who will participate? Who will serve as resource persons? Who will the staff be to support the effort? Where will the meetings be held? Where will those involved stay? What equipment will be needed? What resource materials will be needed? When will the meetings begin? When will they end?

Answers to such questions will depend on a number of factors. Chief among them will be how soon the results must be obtained and how many of the objectives given in the previous section are to be sought. Two important constraints on the approach to be taken are the funding and staffing available for the inquiry. These kinds of factors and constraints are important for deciding whether to undertake a collective inquiry at all and, if that decision is in the affirmative, what approach to take.

In this section, three approaches are described. All of them have been used successfully in actual inquiries. They differ mainly along one dimension, the time required to conduct the inquiry. The first requires about forty-eight hours; the second, about four to six weeks; and the third may take more than a year. It is not intended to imply that inquiries could not take more or less time. These are examples.

For a particular inquiry, combinations of the three might be appropriate. For example. the third approach might include a sequence of inquiries using the first approach. It is intended that the descriptions of these three approaches may help to suggest to the reader how to design her/his own approach to satisfy the need at hand.

1. CHARETTE APPROACH

The term "charette" comes from the early French schools of architecture and refers to the approach used by students for design projects. They worked around the clock during a short period of time to complete their designs.

This approach was adopted by the Forest Service for land management planning for the Shawnee National Forest in Illinois. [3] Simply stated this was a joint effort of Forest Service personnel and concerned citizens to develop the management plan for the Shawnee National Forest through the year 1985. By working together collectively, they exchanged ideas, analyzed the available and potential natural resources, considered the constraints and demands, and developed what they considered to be the best management plan they could. The work began on Friday at 5:30 pm and closed the following Sunday at 5:00 pm. The work continued past midnight for some of the participants.

Although the inquiry only lasted about forty-eight hours, the time required to prepare for it and to process the results was considerably longer. Interested and capable citizens were identified and recruited to participate in the meeting. A study guide was prepared that included salient facts on the issues that would be covered at the meeting. The study guide was mailed to the participants followed by a reminder letter to study the material, attend the meeting, and submit comments on the issues. The comments were analyzed, summarized, and fed back to the participants. Arrangements were made to pre-register the participants by mail. Motel arrangements were made and checked to be sure they were adequate. Audio-visual equipment was obtained and inspected to insure it was working. Arrangements were made and checked to insure there were enough and good meeting rooms, blackboards, chalk, flip-charts, markers, etc. Secretarial services were obtained for the meeting and for processing the results. Resource persons and group leaders were identified, selected, recruited, confirmed, and informed. The group leaders were chosen on the basis they were known to be interested, open minded, and able to encourage broad coverage and participation. Work assignments and schedules were prepared for supporting staff including scribes for each work group. A briefing was given to the supporting staff. Major issues or projects that were to be covered in the Charette were selected, and briefing papers prepared for each of them. Arrangements for registering and welcoming the participants at the meeting were made including assignments to work groups. A speech was prepared for the forest supervisor to give to welcome the participants, and to provide motivation and orientation for their work.

Following the meeting, reference and other materials were retrieved from the participants. Reports from the work groups were collected. The scribes and other relevant staff remained at the site of the Charette to analyze the written reports and to clarify or elaborate on the language when it was deemed desirable. Still at the site, a draft of the summary report was prepared. Later the final report was prepared and disseminated.

Additional arrangements were made for conducting the meetings. Scribes were asked to report on the process of the meetings and to record agreed-upon definitions or contraints, rejected ideas, accepted ideas, and conclusions. Plans were made to minimize interruptions to insure the maximum benefit from the participants' contributions. A plenary session was planned for Sunday afternoon where the group leaders delivered oral reports on the results of their meetings and the forest supervisor thanked the participants for their help.

A summary of the main steps in the Charette approach is given in Table II. Helpful tools for use during such meetings are given in the section on Tools for Collective Inquiries.

TABLE II

CHARETTE PROCESS

1. Prepare a brochure to serve as a study guide for the participants. Include the salient facts on the issues they will address.
2. Identify and recruit relevant participants and mail them the brochure.
3. Send follow up letter to remind participants to study material, attend sessions, and provide comments.
4. Make arrangements to process comments i.e. analyze, summarize, and disseminate.
5. Make arrangements for registering participants by mail.
6. Insure that motel arrangements are in order including audio-visual equipment, auditorium, conference rooms, flip charts, blackboards, secretarial services, computer services and telephone lines, if needed.
7. Select, recruit, confirm, and inform resource persons.

8. Develop list of potential projects to be discussed.
9. Select, recruit, confirm, and inform Group Leaders. Select on basis of personal knowledge of abilities, interest in project area, open-mindedness, and ability to encourage broad coverage and participation.
10. Arrange for registration and reception at meeting. Prepare name tags.
11. Prepare work assignments and schedules for support staff, including Scribes for each work group.
12. Prepare motivation and orientation speech.
13. Give briefing to support staff.
14. Prepare reference materials to be available at meeting.
15. Register participants and assign them to work groups.
16. Deliver motivation and orientation presentation.
17. Introduce Work Group Leaders, Scribes, and resource persons, and break into Work Groups.
18. Insure that Scribes record process, rejected ideas, and accepted ideas.
19. Prevent special-interest "floaters" from traveling from group to group to carry their proposals.
20. Convene final plenary session and make opening remarks on its purpose.
21. Have Group Leaders deliver oral and written reports to top manager.
22. Have top manager thank participants and all others involved for their work.
23. Request participants and others to return reference and other materials and adjourn meeting.
24. Collect registration list and Work Group reports.
25. Hold Scribes and other relevant staff at Charette site to analyze written reports and elaborate on language if needed.
26. At site, prepare draft of summary report.
27. Prepare final report.
28. Distribute final report to participants and other appropriate persons.

2. AT&T/BATTELLE APPROACH

The approach described in this section was first used in Columbus, Ohio, to assess that community's expectations for a public school curriculum. The development of the approach was sponsored by AT&T and Battelle Memorial Institute, Battelle Columbus Laboratory (BCL). [4]

The approach is iterative. Information is requested from participants. The information received is analyzed, categorized, and summarized. The results are then returned to the participants with a request for additional information that elaborates or refines the previous information. A summary report is prepared and distributed.

In the implementation of this approach in Columbus, one of the first steps taken was to obtain the support of the Board of Education for the assessment and to have it appoint a coordinator for the assessment. This was important for conducting the assessment and for the acceptance of the results. Concern for "legitimacy" is important in most collective inquiries.

In developing the approach, a plan was established for recruiting participants for the assessment who represented a cross-section of those who were concerned about the learning outcomes achieved by students. Among those concerned about learning outcomes are students, parents, employers, teachers, and other members of the community. It was important that the plan for obtaining participants emphasized this need for representative participants. There were about 1700 participants (including group leaders) from 14 school areas in the Columbus assessment. The first question used to generate information by the participants was "What does a person leaving high school need to know or be able to do to 'make it' in life?" (This terminology was employed because it was recognized that a number of students leave school before they obtain diplomas.)

Two types of supporting staff were involved, trainers and group leaders. The trainers were graduate students at Ohio State University and the group leaders were teachers, students, parents, businessmen, and other members of the community. The means for selecting group leaders in the 14 school areas is shown in Figure 1. There were 8 trainers and 257 group leaders. The trainers provided training for the group leaders, gave them advice when it was needed, monitored their progress, and helped insure the assessment proceeded on schedule. The group leaders recruited the participants, made arrangements for meeting locations, notified and reminded participants of meeting times, distributed necessary materials at the meeting, led the meetings, collected results, and fed them back to the Battelle staff coordinating the assessment.

	STUDENTS	TEACHERS	COMMUNITY MEMBERS	BUSINESSMEN
DISTRIBUTION OF LEADERS	3 LEADERS REPRESENTING EACH ATTENDANCE AREA	5 LEADERS REPRESENTING EACH ATTENDANCE AREA	2 LEADERS REPRESENTING EACH PUBLIC SCHOOL	12 TO 20 LEADERS REPRESENTING THE ENTIRE COMMUNITY
RESPONSIBILITY FOR SELECTING LEADERS	EACH HIGH SCHOOL PRINCIPAL	EACH HIGH SCHOOL PRINCIPAL	EACH SCHOOL PRINCIPAL	PROJECT DIRECTOR
SELECTION CRITERIA	o JUNIOR OR SENIOR o BOTH SEXES o INTEREST IN TOPIC AND WILLINGNESS TO ORGANIZE OTHER STUDENTS	o FROM FEEDER SCHOOLS AS WELL AS HIGH SCHOOL o BOTH SEXES o INTEREST IN CURRICULUM o FROM VARIETY OF FIELDS	o PARENTS AND NON-PARENTS o BOTH SEXES o NOT EMPLOYED BY CITY SCHOOLS o WILLINGNESS TO SEE AN ACTIVITY THROUGH TO COMPLETION o BASIC READING, WRITING, AND SPEAKING SKILLS	o FAMILIARITY WITH JOB REQUIREMENTS IN HIS BUSINESS o DIRECTLY INVOLVED IN EMPLOYEE TRAINING o REPRESENTATIVES FROM A VARIETY OF TRADES, PROFESSIONS, BUSINESSES, AND SERVICES

FIGURE 1. SELECTION OF GROUP LEADERS

TABLE III

TOPICS COVERED IN THE GROUP LEADER'S GUIDE

(1) WHY IS THIS PROJECT IMPORTANT?

(2) HOW IS THE PROJECT ORGANIZED?

(3) WHAT ARE THE RESPONSIBILITIES OF A GROUP LEADER?

(4) WHAT ARE THE DETAILS ABOUT THE MEETINGS THE GROUP LEADER MUST CONDUCT?

(5) HOW SHOULD THE GROUP LEADER SELECT HER/HIS GROUP MEMBERS? WHAT IS THE "INFORMATION SHEET" THAT IS TO BE DISTRIBUTED? WHAT IS THE "PROJECT OVERVIEW"?

(6) IN ADDITION TO SELECTING AND CONTACTING GROUP MEMBERS, HOW SHOULD THE GROUP LEADER PREPARE FOR THE FIRST MEETING? WHAT ARE SOME OF THE SKILLS THAT AN EFFECTIVE GROUP LEADER POSSESSES? WHAT IS A GOOD WAY TO ARRANGE THE MEETING ROOMS FOR THE COMMUNITY GROUP MEETINGS?

(7) HOW DOES THE GROUP LEADER CONDUCT THE FIRST COMMUNITY MEETING?

(8) WHAT SHOULD A GROUP LEADER DO BEFORE THE SECOND TRAINING SESSION?

(9) WHAT MATERIALS SHOULD A GROUP-LEADER'S TEAM RECEIVE IN THE EQUIPMENT KIT?

(10) HOW SHOULD THE GROUP LEADER PREPARE FOR THE SECOND COMMUNITY MEETING?

(11) HOW DOES THE GROUP LEADER CONDUCT THE SECOND COMMUNITY MEETING? WHAT SHOULD BE REVIEWED FROM SESSION I? HOW DETAILED SHOULD THE SKILL STATEMENTS BE?

(12) WHAT SHOULD THE GROUP LEADER DO AFTER THE TRAINING SESSIONS ARE COMPLETED?

TABLE IV

AT&T/BATTELLE APPROACH FOR ASSESSING COMMUNITY EXPECTATIONS FOR PUBLIC SCHOOL CURRICULUM

1. Obtain the support of the Board of Education and a Coordinator for it.
2. Recruit Trainers and Group Leaders. Develop a plan for recruiting participants.
3. Identify and secure appropriate locations for meetings and schedule them.
4. Prepare materials explaining the goals of the project and the processes to be employed to generate information.
5. Prepare audio tapes to be used by Trainers in training Group Leaders.
6. Develop Group Leaders' Guide.
7. Instruct Trainers on use of audio tapes and Group Leaders' Guide. Discuss anticipated problems Group Leaders might encounter in their community meetings.
8. Prepare instruction for Group Leaders including what their responsibilities are, explanation of how the project is organized, how to recruit community participants, how to arrange location for community meeting, how to prepare for community meeting, what to distribute (posters, handouts, index cards, badges, magic markers, etc.), and the results needed.
9. Insure that Trainers have recruited, confirmed, and reminded Group Leaders and have prepared for and scheduled training sessions.
10. Prepare plan for analysis of meeting results and the scheduling of succeeding meetings.
11. Insure that Group Leaders have recruited representative participants from the community and arranged for meetings.
12. Send postcards to Group Leaders and participants to remind them before each meeting.
13. Insure that Group Leaders report results of meeting to Project Director.
14. Analyze results and prepare feedback to participants and request more refined input.
15. Repeat 12 and 13 and summarize the results.
16. Have participants review results and feed back.
17. Prepare and distribute final report.

The Battelle staff prepared materials for orienting the supporting staff and others. These materials explained the goals of the project and the processes to be employed for generating information. The Battelle staff also prepared audio tapes for the trainers to use in training group leaders and developed a group leader's guide. The topics covered in the guide are shown in Table III. The trainers were instructed on the use of audio tape equipment and the use of the tapes and the group leader's guide. Anticipated problems that the group leaders might encounter were discussed along with possible ways the trainers might suggest for handling them.

The results of the assessment were sent to the 1700 participants, the Board of Education, and the school administration.

A summary of this approach is given in Table IV. Application kits for this approach are available from Communication Technology Corporation in Marlton, New Jersey.

3. WASHINGTON STATE APPROACH

The approach described in this section was used in the State of Washington to help define alternative futures for the state and to formulate proposed legislation to help achieve specific goals. The effort was initiated by the Governor to guide the state government's program planning and to provide criteria for making budget decisions.

There were several categories of participants. First, there was a task force of 165 members. These were chosen from over 4000 nominees as a representative sample of the state's population. Second, there were 1500 participants from ten geographic areas. Third there were 500 from each of these ten geographic areas making a total of 5000. And finally there was a group of 1000.

The initial efforts consisted of four, 3-day workshops lasting 16 hours each day. The participants in these were the 165 members of the task force. The workshops consisted of classroom sessions, in-depth exchanges of information and perceptions, and futures-creating exercises. The 165 produced a number of candidate goals and analyzed them for the impact each would have on the other goals.

These workshops were augmented by areawide meetings in ten geographic areas of the state involving 1500 participants. These meetings were briefer futures-creating sessions.

Following the ten area meetings, the 165 reconvened and finalized eleven alternative futures and a large number of specific goals.

Next an intensive media campaign was mounted to inform the population at large about the results. Weekly television programs were given in an effort to educate citizens. News releases were prepared and distributed to newspapers throughout the state.

Following the media campaign, citizens were surveyed for their preferences. The questions were developed by researchers working with the 165. A telephone survey of 1000 citizens was done to get feedback as quickly as possible. A mail survey was made of 500 citizens in each of the ten geographical areas to determine geographical variations in preferences. The 165 were surveyed by mail to obtain the views of those who had struggled with the issues. And, finally, the 1500 participants in the area conferences were surveyed by mail to get "informed" perspectives.

The results of the surveys were cross-tabulated to provide views of subgroups, e.g. high and low income groups. The results were presented to the 165 task force members who then prepared final recommendations which were strongly influenced by the survey results. There were seven issue areas included: economic growth and population settlement; environmental protection and land use; natural resources and energy; transportation and communication; human development; education and training; and government.

The recommendations were incorporated into the governor's 1975 state of the state address and integrated into 20 of the 30 pieces of legislation he recommended to the legislature.

As may have been noted by the reader, this approach incorporates features of the two approaches described previously.

More detail can be obtained from references [5] and [6].

TOOLS FOR COLLECTIVE INQUIRY

In the first section of this volume, seven steps of a collective inquiry are described beginning with defining the purpose of the inquiry and ending with documenting the process and the results.

In the second section, three approaches to planning the arrangements for a collective inquiry are given. These approaches are especially useful for step two of the first section, i.e. planning the process for the inquiry.

In this section some tools are presented that are intended to be useful in carrying out the other steps. After the tools are described. a report is given on a field test of the tools.

DESCRIPTIONS OF TOOLS

In the pages that follow, brief descriptions of five recommended tools for conducting collective inquiries are given. These are brainwriting, nominal group technique, interpretive structural modeling, voting procedures, and worth assessment. (The options profile is described in Volume 2.) The applicability of these tools and the approaches described earlier are indicated in Figure 2. The descriptions of the tools are intended to give the reader enough information to see their usefulness. On the other hand, those who want to apply the tools will probably need additional information. That can be obtained from the references provided at the end of this volume.

	TOOLS AND APPROACHES									
STEPS	Shawnee Forest Charette	AT&T/BCL School Curricula	Goals for Washington State	Options Profile	Nominal Group Technique	Brainwriting	ISM Program	ISM Amendment	Worth Assessment	Voting Procedures
Define the Purpose(s)					X	X				
Plan the Inquiry Process	X	X	X							
Generate the Elements of the Issue					X	X				
Choose a Contextual Relation					X	X				
Determine the Relations Among the Elements							X	X		X
Analyze, Amend, and Rank the Results				X				X	X	X
Document the Process and the Results				X						

Figure 2. COLLECTIVE INQUIRY PROCEDURES*

*An X in a cell indicates that the Tool or Approach at the top is applicable to the Step in the row. A blank cell does not mean that the Method is not applicable.

BRAINWRITING (IDEAWRITING)

A collective inquiry tool useful for generating ideas by small groups. Ideas are generated by individuals stimulated by a carefully prepared, trigger question. Ideas are written on sheets of paper. The sheets are passed to other participants who elaborate or clarify the ideas already recorded or add new ideas. This tool is helpful in defining problems, conceptualizing approaches to solutions, and formulating policies. It is especially useful when uncertainty or controversy exists about an issue or problem and its possible resolution. It works well even when it is important to neutralize the effects of dominant individuals.

APPROPRIATE CONDITIONS FOR USE

- o A need to collect ideas relevant to some issue.
- o A carefully prepared, focused trigger question.
- o Qualified persons willing to participate.
- o Qualified group leader willing to facilitate.

APPLICATION AREAS

- o Generally applicable when there is a need for collective idea generation. Especially useful for defining problems, requirements, or objectives to be used as inputs for other tools, e.g. ISM or worth assessment.
- o Useful for involving stakeholders in planning.

RESULTS OF APPLICATION

- o A list of 50 to 100 ideas about some issue.
- o Increased understanding by the participants about the issue, the ideas, and each other.
- o An opportunity for all to contribute ideas.

RESOURCES REQUIRED FOR APPLICATION

- o No more than six persons for a single group. Multiple groups may work simultaneously.
- o Each group needs a quiet place to work, a table and chairs, paper, and pencils.
- o From 15 minutes to two hours for process.
- o Funds, if required, for participants or leader.

HOW THE PROCESS WORKS

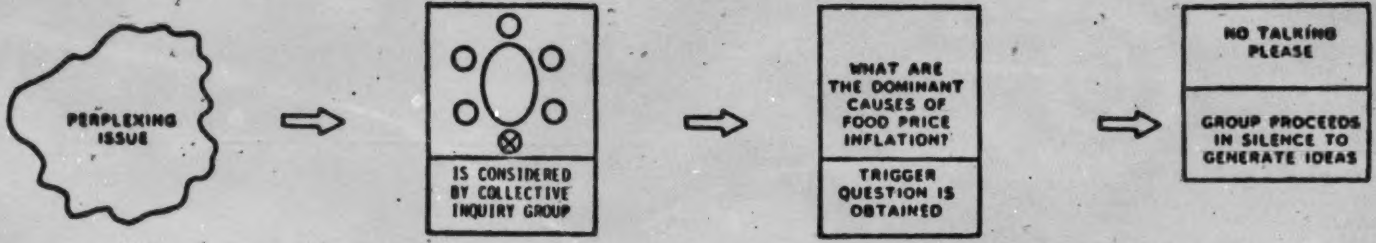
- o First, silent generation of written ideas by individual participants stimulated by carefully prepared, focused trigger question written at the top of a sheet of paper.
- o After about 5 to 10 minutes, participants exchange sheets of paper and build on the ideas already recorded or add new ideas that occur.
- o Process continues with further exchanges until all or most of the participants have seen all or most of the sheets.
- o Participants or facilitator edits the lists by clarifying and coalescing similar ideas.

IMPORTANT ATTRIBUTES OR FEATURES

- o Potential for generating many ideas concerning organizational or behavioral issues.
- o Potential for encouraging contributions from reticent participants or moderating dominant ones.
- o Opportunity for all stakeholders in issue to provide inputs to the process.

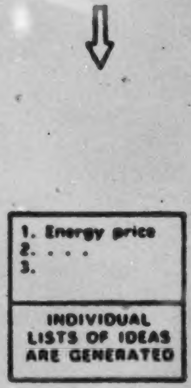
RELATED TOOLS

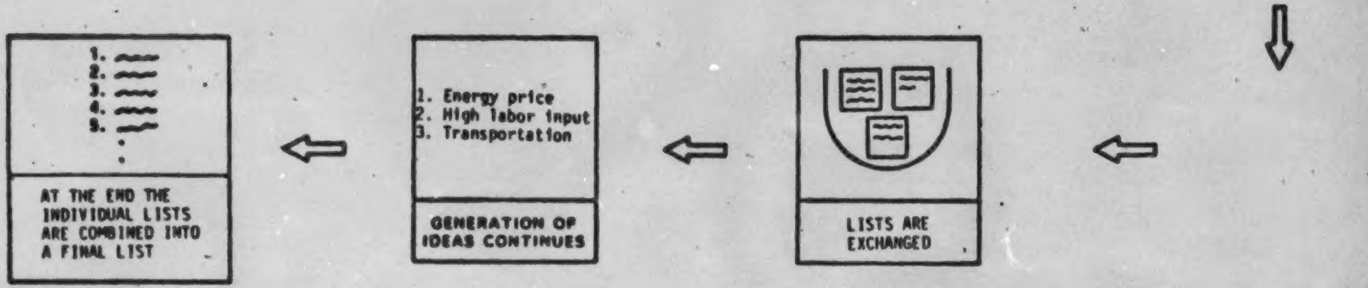
- o No other tools are required to use brainwriting.
- o Alternate tools are nominal group technique, Delphi, and content analysis.
- o Results may be used as input to ISM or worth assessment.


PERPLEXING ISSUE
IS CONSIDERED BY COLLECTIVE INQUIRY GROUP
WHAT ARE THE DOMINANT CAUSES OF FOOD PRICE INFLATION?
TRIGGER QUESTION IS OBTAINED
NO TALKING PLEASE
GROUP PROCEEDS IN SILENCE TO GENERATE IDEAS

BRAINWRITING

Faced with the need to generate ideas related to an issue or problem, a facilitator is obtained, facilities are obtained, and a group knowledgeable about the issue is convened. A carefully prepared, focused trigger question is phrased by or explained to the group. Stimulated by the question, the members write their ideas on sheets of paper. After 5 to 10 minutes, the lists of ideas are exchanged among the members and each tries to elaborate or clarify the ideas on the sheet or adds new ideas that occur to him. The process continues until a set time is reached or the group has no new ideas or the sheets have been passed to everyone. Then the ideas are clarified and/or coalesced to produce a final list.


1. Energy price
2. . . .
3.
INDIVIDUAL LISTS OF IDEAS ARE GENERATED


LISTS ARE EXCHANGED
1. Energy price
2. High labor input
3. Transportation
GENERATION OF IDEAS CONTINUES
1.
2.
3.
4.
5.
AT THE END THE INDIVIDUAL LISTS ARE COMBINED INTO A FINAL LIST

NOMINAL GROUP TECHNIQUE (NGT)

A collective inquiry tool useful for generating ideas by small groups. Ideas are generated by individuals, then discussed, clarified, elaborated, and combined by the group. This tool is helpful in defining problems, conceptualizing approaches to solutions, and formulating policies. It is especially useful when uncertainty or controversy exists about an issue or problem and its possible resolution. It works well even when it is important to neutralize the effects of dominant individuals to allow all participants to contribute ideas.

APPROPRIATE CONDITIONS FOR USE

- A need to collect ideas relevant to some issue.
- A carefully prepared, focused trigger question.
- Qualified persons willing to participate.
- Qualified group leader willing to act as a facilitator; preferably not an expert on the issue.

APPLICATION AREAS

- Generally applicable when there is a need for collective idea generation. Especially useful for defining problems, requirements, or objectives to be used as inputs for other tools, e.g. ISM or worth assessment.
- Useful for involving stakeholders in planning.

RESULTS OF APPLICATION

- A list of 20 to 100 ideas about an issue.
- A preliminary ranking of the ideas according to a chosen criterion.
- An opportunity for all to contribute ideas.

RESOURCES REQUIRED FOR APPLICATION

- 6 to 10 persons for a single group. Multiple groups may work simultaneously.
- Each group needs a quiet place to work, a table and chairs, paper and pencils. Leader needs a flip chart and felt-tip pen plus a place to tape up sheets.
- One to two hours time for process.
- Funds, if required, for participants or leader.

HOW THE PROCESS WORKS

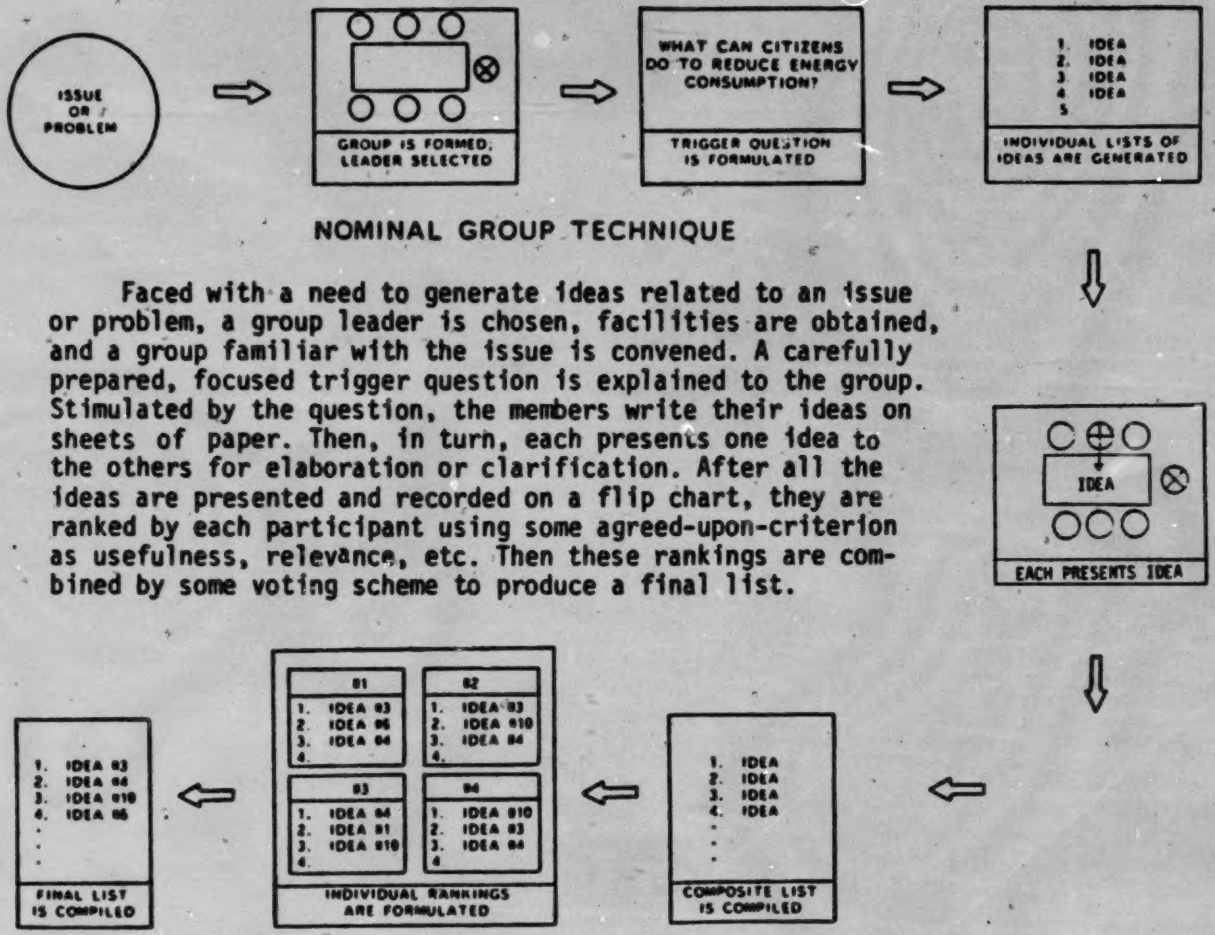
- First, silent generation of written ideas by individual participants stimulated by an oral presentation of a carefully prepared, focused trigger question.
- Individuals present ideas one at a time round robin.
- Spontaneous elaboration or clarification of ideas is encouraged, but no discussion or criticism.
- Ideas are recorded on flip chart to form group list.
- Round robin discussion of the resulting list.
- Voting on the ranking of the ideas generated.

IMPORTANT ATTRIBUTES AND FEATURES

- Potential for generating many ideas concerning organizational or behavioral issues.
- Potential for encouraging contributions from reticent participants and moderating dominant ones.
- Opportunity for all stakeholders in issue to provide inputs to the process.

RELATED TOOLS

- No other tools are required to use NGT.
- Alternative tools are brainwriting, Delphi, and content analysis.
- Any of several voting schemes may be used with NGT.
- Results may be used as input to ISM or worth assessment.


ISSUE OR PROBLEM
GROUP IS FORMED, LEADER SELECTED
WHAT CAN CITIZENS DO TO REDUCE ENERGY CONSUMPTION?
TRIGGER QUESTION IS FORMULATED
1. IDEA
2. IDEA
3. IDEA
4. IDEA
5
INDIVIDUAL LISTS OF IDEAS ARE GENERATED
IDEA
EACH PRESENTS IDEA
1. IDEA
2. IDEA
3. IDEA
4. IDEA
COMPOSITE LIST IS COMPILED
#1
1. IDEA #3
2. IDEA #6
3. IDEA #4
4.
#2
1. IDEA #3
2. IDEA #10
3. IDEA #4
4.
#3
1. IDEA #4
2. IDEA #1
3. IDEA #10
4.
#4
1. IDEA #10
2. IDEA #3
3. IDEA #4
4
INDIVIDUAL RANKINGS ARE FORMULATED
1. IDEA #3
2. IDEA #4
3. IDEA #10
4. IDEA #6
FINAL LIST IS COMPILED

NOMINAL GROUP TECHNIQUE

Faced with a need to generate ideas related to an issue or problem, a group leader is chosen, facilities are obtained, and a group familiar with the issue is convened. A carefully prepared, focused trigger question is explained to the group. Stimulated by the question, the members write their ideas on sheets of paper. Then, in turn, each presents one idea to the others for elaboration or clarification. After all the ideas are presented and recorded on a flip chart, they are ranked by each participant using some agreed-upon-criterion as usefulness, relevance, etc. Then these rankings are combined by some voting scheme to produce a final list.

INTERPRETIVE STRUCTURAL MODELING (ISM)

A computer assisted learning process than enables an individual or a group to develop a structure showing interrelations among a set of given elements.

APPROPRIATE CONDITIONS FOR USE

- A set of elements related to the issue.
- A relationship which is appropriate to interrelate the elements.
- A complex issue requiring understanding of interactions among many elements.
- Persons who have knowledge of the issue and are willing to participate.

APPLICATION AREAS

- Generally applicable where there is a need to relate elements of an issue, problem, system, etc. to obtain a holistic view.
- Useful as a step toward the development of quantitative models
- Useful for setting priorities or precedences.

RESULTS OF APPLICATION

- A structured model of related elements.
- A carefully refined language to describe or discuss an issue or system.
- A shared and improved understanding and definition of elements and relations used.
- Enhanced understanding of the whole.

RESOURCES REQUIRED FOR APPLICATION

- A responsible client for the study who wants to benefit from knowledge of available participants concerning a particular issue.
- Up to 8 willing and able participants, a group leader familiar with interpretive structural modeling, and a computer operator.
- A time-shared computer with programs for structuring. A large screen display is helpful, but an exercise may be conducted using a teletypewriter-type terminal with auxiliary small screen television display.

RESOURCES REQUIRED (continued)

- Funds for participants' and leader's time; equipment, approximately $50 per hour; and telephone lines.
- The time required for an exercise depends upon the number of elements in the model and their complexity. A ten-element exercise might take one to two hours.

HOW THE PROCESS WORKS

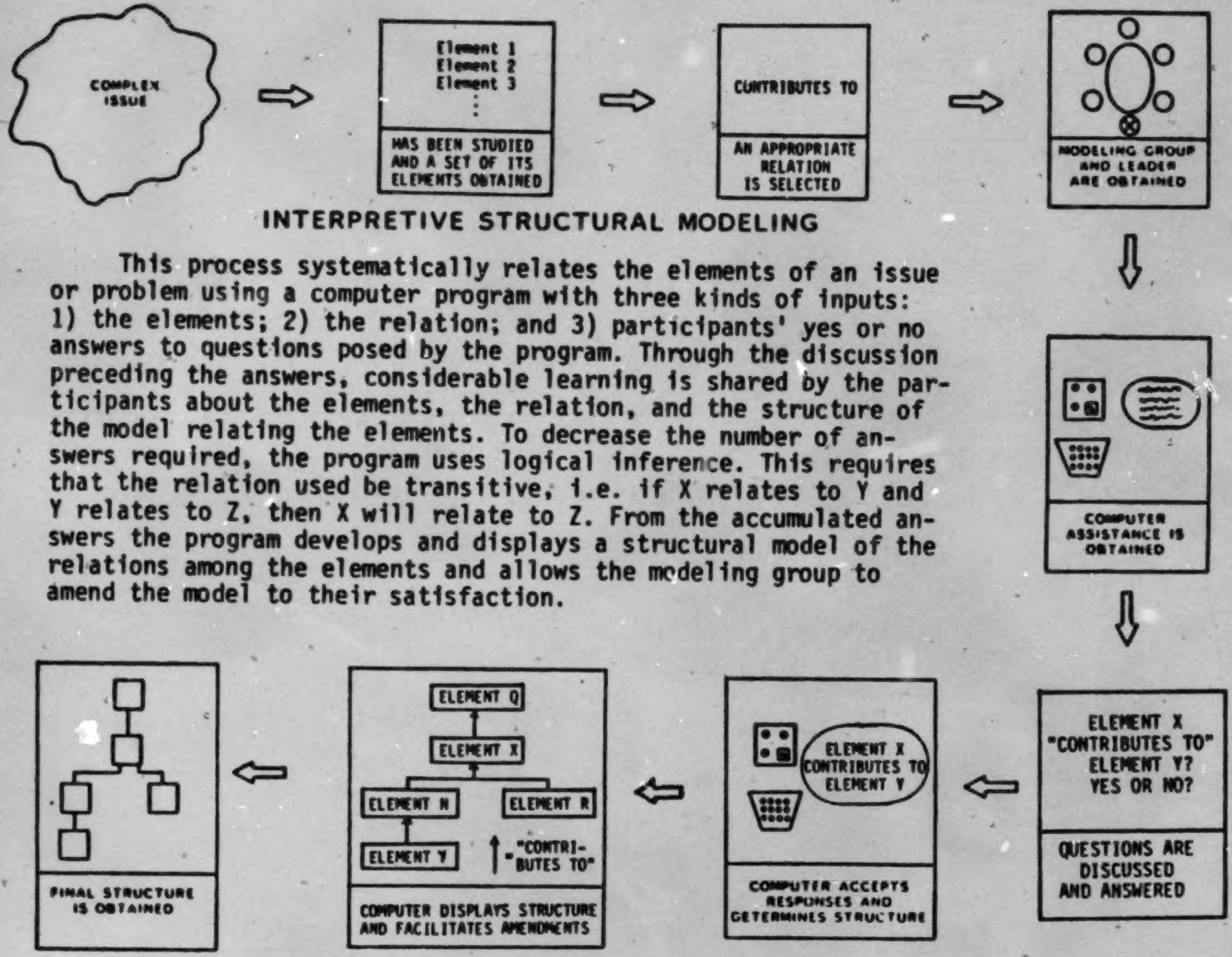
- An issue and structuring theme are identified.
- A group and a process leader are chosen.
- Elements and contextual relation are obtained.
- The group responds to computer-posed questions relating elements.
- The computer displays the structure developed.
- The group amends the structure until it is satisfactory.

IMPORTANT ATTRIBUTES AND FEATURES

- Appropriate relation must be chosen carefully.
- Elements and relation are clarified by reasoning and discussion stimulated by the process.
- The quality of the results obtained is strongly dependent upon the leader. He should facilitate and not impose his knowledge of the issue.
- Overemphasis of the mechanistic and technical aspects of the process should be avoided.

RELATED TOOLS

- Nominal group technique, brainwriting, Delphi, literature search, or a combination of these can generate elements and contextual relations for the process.
- In simple situations, "rearrange and tape" and other heuristic, non-computer-assisted methods may be used. Great care must be taken to avoid overly simple, erroneous results using these approaches.
- The process may be used as a step to quantitative modeling or to structuring attributes or assessing worth for making choices.


COMPLEX ISSUE
Element 1
Element 2
Element 3
HAS BEEN STUDIED AND A SET OF ITS ELEMENTS OBTAINED
CONTRIBUTES TO
AN APPROPRIATE RELATION IS SELECTED
MODELING GROUP AND LEADER ARE OBTAINED
COMPUTER ASSISTANCE IS OBTAINED
ELEMENT X "CONTRIBUTES TO" ELEMENT Y? YES OR NO?
QUESTIONS ARE DISCUSSED AND ANSWERED
ELEMENT X CONTRIBUTES TO ELEMENT Y
COMPUTER ACCEPTS RESPONSES AND DETERMINES STRUCTURE
ELEMENT Q
ELEMENT X
ELEMENT N
ELEMENT R
ELEMENT Y
= "CONTRIBUTES TO"
COMPUTER DISPLAYS STRUCTURE AND FACILITATES AMENDMENTS
FINAL STRUCTURE IS OBTAINED

INTERPRETIVE STRUCTURAL MODELING

This process systematically relates the elements of an issue or problem using a computer program with three kinds of inputs: 1) the elements; 2) the relation; and 3) participants' yes or no answers to questions posed by the program. Through the discussion preceding the answers, considerable learning is shared by the participants about the elements, the relation, and the structure of the model relating the elements. To decrease the number of answers required, the program uses logical inference. This requires that the relation used be transitive, i.e. if X relates to Y and Y relates to Z, then X will relate to Z. From the accumulated answers the program develops and displays a structural model of the relations among the elements and allows the modeling group to amend the model to their satisfaction.

VOTING

Voting is a prevalent method of democratic group decision making. Any voting system can lead to unintended or paradoxical results where preferred candidates lose. In certain situations certain voting algorithms are better than others. It is helpful to be able to identify which voting systems are appropriate for given situations.

APPROPRIATE CONDITIONS FOR USE

- o Candidates and their impacts have been determined and a democratic method is needed to select a single alternative.
- o A group needs to make a decision concerning selection of an alternative.
- o Group consensus to select an alternative is not easily obtainable.
- o A legal mandate for voting exists.

APPLICATION AREAS

- o Generally applicable in all democratic situations when a group must choose among competing candidates.

RESULTS OF APPLICATION

- o A winner is determined as well as an indication concerning the strength of preference of the group for the winner.

RESOURCES REQUIRED

- o A set of candidates and a set of voters.
- o A mandate or desire to decide democratically.
- o A leader to explain the procedure and tally the votes.

HOW THE PROCESS WORKS

- o There are many voting systems. Some useful ones are:
  - Plurality: Voters vote for one candidate and the one receiving the most votes wins.
  - Majority Rule: Voters vote for one candidate. A candidate must receive more than 50% of the votes to win. If no candidate wins, runoff elections are held among candidates whose combined votes in the previous election constitute a plurality.
  - Weighted Voting: Voters are given a fixed number of votes. These are assigned by voters to competing candidates according to their strength of preference. The winner is the one with the greatest point total. In one such system with N candidates the most preferred is given N-1 votes, the second most preferred N-2 votes, etc. Alternately, voters may assign their own preference votes on a fixed end point scale. This method of voting, also called Borda voting, is particularly vulnerable to "strategic" voting, in which people do not vote for candidates according to their preferences in order to insure victory for their preferred candidate.
  - Binary Comparison Voting: Voters note their preference ordering among all candidates or vote in a binary fashion for all possible paired alternative combinations. In one method, known an Condorcet voting, the candidate that beats all others in pairwise contests is the winner. In another method, known as Copeland voting, a candidate's score equals the number of candidates which it defeats by simple majority voting minus the number that defeat it. The candidate with the highest point total wins. A Condorcet winner will also be a Copeland winner. If there is no Condorcet winner, the Copeland method selects the candidate that has the greatest win-minus-loss score.

IMPORTANT ATTRIBUTES AND FEATURES

- o Potential for developing recommendations of preferred candidates.
- o Potential for voting paradoxes.

RELATED TOOLS

- o Various tools which require elements for inputs use voting. It is especially appropriate to precondition voters for informed voting.
- o Worth Assessment is an alternative approach.
- o Informed discussion which may be assisted by many of the collective inquiry tools presented in this volume may lead to informed voting and perhaps even consensus.

ALTERNATIVE
SELECTION BY A
GROUP IS NEEDED

DISCUSSION
DEBATE
DELPHI
BRAINWRITING

INFORMED DISCUSSION
IS USED TO ENHANCE
VOTING ON THE ISSUE
UNDER CONSIDERATION

PLURALITY
MAJORITY RULE
WEIGHTED VOTING
BINARY COMPARISON
VOTING

A VOTING PROCEDURE
IS SELECTED

VOTING

A procedure in which a group expresses its preference for an alternative from among competing alternatives. Many voting procedures exist and the competing concerns involving trade-offs between simplicity and ability to detect voting paradoxes should be considered in selecting a voting procedure.

- Group X - 500 voters prefer E to D to A to C to B
- Group Y - 400 voters prefer B to C to E to A to D
- Group Z - 200 voters prefer D to A to B to C to E

VOTES ARE TALLIED
ACCORDING TO THE
SCHEME SELECTED

- Plurality Voting - E Wins
  (E has 500 votes, B has 400, D has 200)
- Majority Voting - B Wins
  (In runoff election between B and E, E has 500 votes, B has 600)
- Condorcet Voting - A Cyclical Majority Exists and there is no winner
- Copeland Voting - D Wins
  A beats B,C: B beats C,E: C beats E, D beats A,B,C: E beats D

VOTES ARE ANALYZED

WHO
WINS?

THE WINNER
IS
SELECTED

WORTH ASSESSMENT

Evaluation or selection of activities or alternatives is often based on subjective estimates of their worth. Worth assessment is a collective inquiry tool for translating qualitative impressions of value into understandable, consistent, and meaningful quantitative evaluations. This is accomplished by identifying and organizing the attributes of the activities or alternatives into a worth structure. The attributes are assigned weights according to their importance to the final result. Each alternative is then valued using the weighted worth structure to provide a quantitative basis for decision making.

APPROPRIATE CONDITIONS FOR USE

- A need to decide among alternatives.
- A need to evaluate alternatives quantitatively.
- A need to predict individual's decisions.
- A need to make individual or group values explicit.
- A need to communicate the implications of alternatives.
- A need to consider and organize multiple objectives and assessment criteria.

APPLICATION AREAS

- Generally applicable for decision making by helping to specify, interpret, or value the impacts of alternatives on individuals of groups.

RESULTS OF APPLICATION

- A tree structure showing lower level criteria (attributes or objectives) which compose higher level criteria.
- An easily communicated display of the value or importance individuals or groups place upon criteria for proposed alternatives.
- An explicit valuation of proposed alternatives according to the importance placed on their attributes.

RESOURCES REQUIRED FOR APPLICATION

- A set of alternatives for a decision.
- An individual or group that wants them evaluated.
- Qualified persons to identify and weight criteria.
- One to ten hours of participants' time.
- A qualified leader.

HOW THE PROCESS WORKS

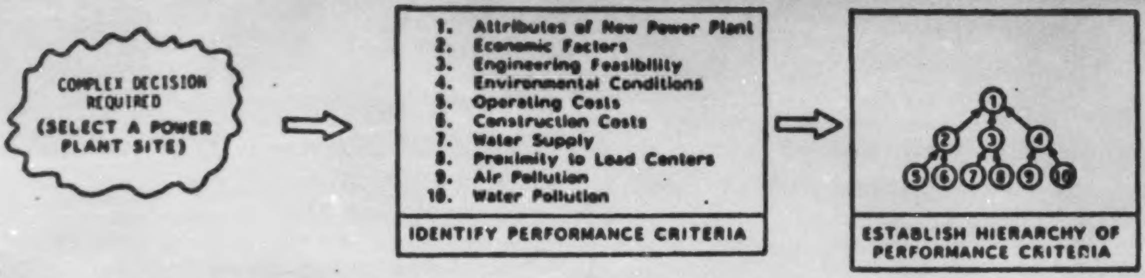
- The individual or group whose subjective values are to be assessed is identified.
- The alternatives to be evaluated are identified.
- A set of performance criteria (attributes or objectives) are arranged in a tree structure such that lower criteria are parts of the criteria above them.
- The criteria are assigned weights according to their relative importance (worth) to give a worth structure.
- Each alternative is evaluated using the worth structure.
- The values of the alternatives are compared for making decisions.

IMPORTANT ATTRIBUTES AND FEATURES

- An explicit procedure that incorporates human judgement and values into a quantitative assessment.
- Multiple worth connections among assessment criteria and their consequences on performance can be shown.
- Risk considerations can only be approximated.
- Sensitivity analysis can be done to show the effects of the weights assigned.

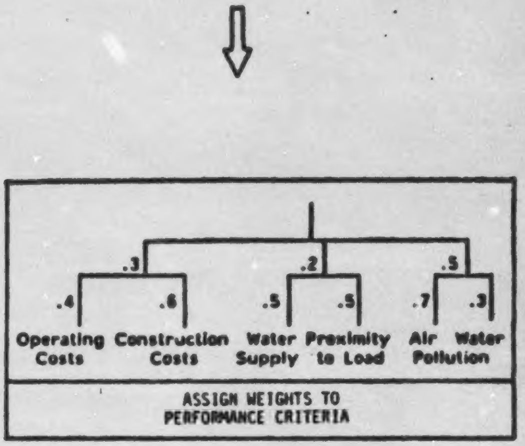
RELATED TOOLS

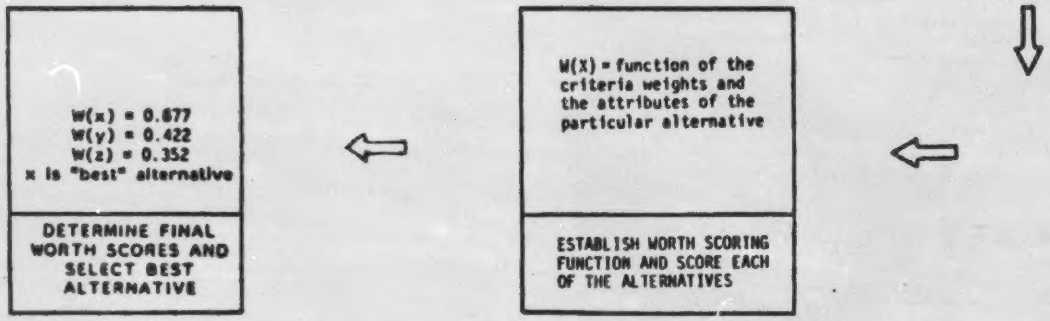
- Brainwriting, NGT, or Delphi may be used to identify criteria (attributes or objectives).
- Interpretive structural modeling is useful for developing the worth structure.
- Decision analysis is an alternative tool, especially when probabilistic considerations are important.


COMPLEX DECISION REQUIRED
(SELECT A POWER PLANT SITE)
1. Attributes of New Power Plant
2. Economic Factors
3. Engineering Feasibility
4. Environmental Conditions
5. Operating Costs
6. Construction Costs
7. Water Supply
8. Proximity to Load Centers
9. Air Pollution
10. Water Pollution
IDENTIFY PERFORMANCE CRITERIA
1
2
3
4
5
6
7
8
9
10
ESTABLISH HIERARCHY OF PERFORMANCE CRITERIA

WORTH ASSESSMENT

There is a need to choose among alternatives proposed to resolve an issue or problem. A set of performance criteria is established. The criteria are structured into a worth hierarchy and weights are assigned to the criteria according to their importance to the overall issue or problem. The worth of each alternative is valued using the weights and its particular attributes. The values for all alternatives are compared for making the final decision.


.3
.2
.5
.4
.6
.5
.5
.7
.3
Operating Costs
Construction Costs
Water Supply
Proximity to Load
Air
Water
Pollution
ASSIGN WEIGHTS TO PERFORMANCE CRITERIA


W(x) = 0.677
W(y) = 0.422
W(z) = 0.352
x is "best" alternative
DETERMINE FINAL WORTH SCORES AND SELECT BEST ALTERNATIVE
W(X) = function of the criteria weights and the attributes of the particular alternative
ESTABLISH WORTH SCORING FUNCTION AND SCORE EACH OF THE ALTERNATIVES

FIELD TEST OF TOOLS

After the tools described above had been chosen as candidates for assisting the conduct of collective inquiries, a field test was conducted. The purposes of the test were to determine if the descriptions of the tools were adequate and whether the tools were useful.

To accomplish the test, it was desirable to find panelists who would be interested in participating and who would be qualified to test the tools. The decision was made to ask the Tennessee Valley Authority to participate. It has a large environmental education program. It is responsible for regional development. And, it is very much interested in and involved in conducting collective inquiries. TVA consented and ten of its staff participated in the testing: 4 from environmental education; 2 from community development; 2 from the director's office; 1 from the citizen action office; and 1 from natural resources. All were involved with collective inquiries in one way or another.

The tools that were tested were: brainwriting; nominal group technique; worth assessment; voting procedures; and interpretive structural modeling. For each of these tools, the panelists were asked to judge the pictorial and written information describing the tool. These evaluations of the materials were use to make improvements in the descriptions.

For brainwriting, nominal group technique, and interpretive structural modeling, the participants made trial applications of each on real issues of importance to TVA. These applications were evaluated by the participants' responses (using a five point scale) to the following questions.

Q1. Was the method difficult to use? 1: very difficult; 5: easy.

Q2. Did the results achieved justify the resources used? 1:too expensive; 5: well worthwhile.

Q3. Did the results achieved justify the time of the participants? 1: wasteful; 5: well worthwhile.

Q4. As a participant, did you like the method? 1: disliked it; 5:liked it very much.

Q5. Was the application of the method effective? 1: not effective; 5: very effective.

For brainwriting, the averages of the evaluations for each question were as follows. Q1:3.6; Q2:3.7; Q3:3.9; Q4:3.9; Q5:3.4.

For the nominal group technique, the averages were: Q1:4.5; Q2:4.4; Q3:4.4; Q4:4.6; Q5:4.7.

For interpretive structural modeling, the averages were: Q1:4.3; Q2:3.7; Q3:3.8; Q4:4.0 Q5:4.1.

In addition to providing the numerical ratings, the participants also made a number of useful suggestions for improvements.

Although no statistical significance can be given to the results (lack of time, money, and appropriate panelists prevented such results), they were useful for a number of reasons. The suggestions made for improving the pictorial and written materials were very helpful. The panelists were well qualified, interested, and openly skeptical when they started. Their evaluations, especially of the applications, supported the judgments of the project team that these were useful tools for conducting collective inquiries. Moreover, applications of combinations of tools proved useful and indicated some improvements in ways to combine tools.

In summary, the field test supported the choice of tools and provided useful information for improving them.

In addition to the field test of the tools described above, another field test was conducted on the options field described in Volume 2. Briefly, this tool is designed to help in collective inquiries aimed at the design of environmental learning systems.

The participants in this field test were four TVA staff and twelve staff from colleges and universities that work with TVA in environmental education.

This test was conducted to obtain the judgements of experienced environmental educators on the options field as a design tool. The participants were given two questions at the beginning that they would be asked to answer at the end. These questions were:

1. What are the pros and cons of the design process (using the options field)?

2. What are the pros and cons of the dimensions and the options?

To answer these questions, the participants needed a "hands-on" experience using the design process. To that end they were asked to participate in a design exercise. The first part of the exercise was to develop a restriction structure. Detailed information is provided in Volume 2, but stated briefly this means the following. In the design of an environmental learning system, choices of options in one dimension, say, Presumed Learner Skills Base can restrict the choice of options in another dimension, say, Basic Learning Outcomes Sought. To develop the restriction structure, the participants used an interpretive structural modeling (ISM) approach. The elements were:

A. PRESUMED LEARNER SKILLS BASE

B. PRESUMED LEARNING STYLE

C. SOURCE OF INFORMATION

D. MODE OF ENVIRONMENTAL EDUCATION

E. BASIC LEARNING OUTCOMES DESIRED

F. TYPE OF ENVIRONMENTAL EDUCATION

G. CURRICULUM DELIVERY CONCEPT

H. MEDIATOR MODEL

I. LEARNER INTERACTION RESOURCES

J. ORIGIN OF FINANCING

The relation used was: "A choice of option(s) in Dimension X can restrict the choice of options in Dimension Y".

Through their participation in this exercise to design the restriction structure, the participants gained familiarity with the dimensions and the options within the dimensions.

After the restriction structure was developed, the participants participated in a second ISM exercise to develop a precedence structure. This structure ordered the dimensions of the options field for selection of options. The elements of this exercise were dimensions, and the relation used was: "A choice of option(s) in Dimension X should take priority over a choice in Dimension Y".

After the precedence structure was completed, the participants began (but did not finish because of time) to design a regional environmental learning system. The purpose was to involve the participants in the use of the design process so that they would be able to evaluate it.

The first question listed above was addressed first. To answer it a nominal group technique was used with the following questions: "What are the positive features of the design process?" and "What are the limitations of the design process?" Items given in response to the first question (with no priority implied) were:

- Flexible; adaptable to region; can be modified
- Comprehensive
- Gives direction, structure for eliciting decisions; systematic
- Offers opportunity for different views and disciplines to be included in the design
- Leads to consensus decisions which should lead to lasting effects. Includes realities of compromise
- Engenders thinking and analysis
- Involves individuals in the process
- Illustrates design and dimensions of the design - enables review by the group
- Permits computer-aided design
- Forces distinguishing between synthesis and analysis
- Useful for capturing random ideas into useable form
- Builds on experience and, hopefully, encourages ingenuity

Items given in response to the question on limitations (with no priority implied) were:

- Limited population with ability to use process, limited ability to think through process
- Restriction concept needs to be presented more clearly
- Requires individuals who are interested and willing to give their time
- Time to use right limits its use. Needs to be refined
- Parameters (scenario) needs to be defined at start
- Understanding dimensions' and options' denotations and connotations is very hard
- Paramount to use well-organized facilitator training procedure
- Lack of understanding of systems concepts affects people's feelings about process. Creates hostility, opting out of process.
- Needs to be made clear that process is iterative
- Outcome of process is affected by group interaction skills of participants
- Facilitator needs to lay out work, pace work, and take stock of progress periodically

Time did not permit the second question regarding dimensions and options to be addressed in the same way as the first. Instead, the participants individually submitted lists of pros and cons regarding the dimensions and the options including some suggestions for additional options. Many of these have been incorporated in the material presented in Volume 2.

In summary, the options field is judged to be a potentially useful tool in the design of environmental learning systems but must be used with qualified participants and facilitators, and the context should be clearly defined.

REFERENCES

1. J.N. Warfield, "TOTOS: Improving Group Problem Solving", Approaches to Problem Solving, Number 3, The Academy for Contemporary Problems, Columbus, Ohio , 1975.

2. J.N. Warfield, "Methods of Idea Management", Approaches to Problem Solving, Number 4, The Academy for Contemporary Problems, Columbus, Ohio, 1975.

3. Lowell W. Patterson, "Shawnee National Forest Land Management Charette", U.S. Forest Service, U.S. Department of Agriculture, Washington, D.C., 1977.

4. K.S. Keller, "Development and Implementation of a Methodology for Assessing Community Expectations for Public School Curriculum", Battelle Monograph No. 5, April 1974.

5. Alternatives for Washington State-Wide Task Force, "Citizens' Recommendations for the Future", Vol. 1, Olympia, Washington, Office of Program Planning and Fiscal Management, 1975.

6. John M. Wardwell and Don A. Dillman, "The Alternatives for Washington Surveys: The Final Report", Vol. 6, Olympia, Washington, Office of Program Planning and Fiscal Management, 1975.

COLLECTIVE INQUIRY BIBLIOGRAPHY

L. Adelman, T.R. Steward, & K. Hammond, "A Case History of the Application of Social Judgment Theory to Policy Formulation," Policy Science, Vol. 6, 1975.

Advisory Commission on Intergovernmental Relations, Citizen Participation in Fiscal Decision-Making at the Federal, State, and Local Levels. (Draft)

R. Behn & J. Vaupel, "Decision Making for Sophomores," Urban Analysis, Vol. 3, 1976.

D.M. Conner, Citizens Participate: An Action Guide For Public Issues. Oakville, Ontario: Development Press, 1974.

Andre L. Delbecq, Andrew H. Van de Ven, and David H. Gustafson, Group Techniques for Program Planning: A Guide to Nominal Group and Delphi Processes, Scott, Foresman and Company, Glenview, Illinois, 1975.

D. Dillman, "Preference Surveys and Policy Decisions: Our New Tools Need Not Be Used in the Same Old Way," Journal of Community Development Society, Vol. 8, Spring, 1977.

D. Dillman & K. Tremblau, "The Quality of Life in Rural America," The Annals of the American Academy, January 1977.

P.S. Elder, Environmental Management and Public Participation. Ontario: Canadian Environmental Law Research Foundation, 1975.

R. Fitz & J. Troha, Dialogue, Decision, Action, Evaluation: A Workbook for Program Planning, Dayton, Ohio, Mariansit Training Network, 1977.

K. Hammond, Risks and Safeguards in the Formulation of Social Policy. Boulder: Center For Research on Judgment and Policy, Institute of Behavioral Policy, 1978.

K. Hammond, "Toward Increasing Competence of Thought in Public Policy Formation," Public Policy Formation, 1977.

K. Hammond & L. Adelman, "Science, Values and Human Judgment," Science, October 22, 1976.

K. Hammond, J.L. Mumpower, & T.H. Smith, "Linking Environmental Models with Models of Human Judgment: A Symmetrical Decision Aid," IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-7, No. 5, May, 1977.

K. Hammond, J. Rohrbaugh, J. Mumpower, & L. Adelman, "Social Judgment Theory: Applications in Policy Formation," Human Judgment and Decision Process in Applied Settings.

J.D. Hendee, et al., *Public Involvement and the Forest Service: Experience, Effectiveness, and Suggested Direction: A Report from the U.S. Forest Service Administrative Study of Public Involvement.* Washington, D.C.: U.S. Forest Service, 1973.

R.W. House, "Applications of Interpretive Structural Modeling in Brazil's Alcohol Fuel Program", *IEEE Transactions on Sytems, Man and Cybernetics*, SMC 9(7), July 1979.

R.W. House, C.H. Kimzey, and R.T. Nahs, "On Developing Policies to Improve Technological Innovation and Productivity in the United States", *IEEE Transactions on Systems, Man, and Cybernetics*, SMC 9(9), September 1979.

K. Kawamura, D.H. Brand, Jr. and D.M. Irvin, "Implementation of Interpretive Structural Modeling in a State-Level Planning Context," *Modeling and Simulation Proceedings of the Seventh Annual Pittsburgh Conference*, Vol. 7, Part 2, April 26-27, 1976.

K.S. Keller, "Development and Implementation of a Methodology For Assessing Community Expectations For Public School Curriculum," *Battelle Monograph*, April, 1974, (Number 5).

R.M. Lake, ed., *Forest Service Inform and Involve Handbook (Draft)*. Washington, D.C.: U.S. Forest Service, U.S. Dept. of Agriculture, 1973.

B.W.G. Marley-Clarke, "Policy Planning for Environmental Management," *Long Range Planning*, October, 1976.

R.G. Niemi & W.H. Riker, "The Choice of Voting System," *Scientific American*, June, 1976.

H. Nix, *"The Community and Its Involvement in the Study, Planning, Action Process"*. HEW Publication CDC 78-8355.

L.W. Patterson, *Shawnee National Forest Land Management Planning, Charette* (draft). Washington, D.C.: U.S. Forest Service, U.S. Dept. of Agriculture, 1977.

J. Pierce, *Water Politics and Public Involvement*. Ann Arbor: Ann Arbor Science, 1976.

J. Rohrbaugh, "Cognitive Maps: Describing The Policy Ecology of a Community" *Great Plains-Rocky Mountain Geographical Journal*, Vol. 6, No. 1, April, 1977.

J.B. Rosenberg,"A Cafeteria of Techniques and Critiques," *Public Management*, Vol. 57, December, 1975.

A.P. Sage, et al., *User's Guide to Public Systems Methodology* (partial draft). NSF (AER 77-16865), University of Virginia, 1978.

A.P. Sage, Methodology for Large-Scale Systems, McGraw-Hill Book Company, New York, New York, London, Tokyo, Toronto, 1977.

Southern Rural Development Center, "Citizen Participation in Rural Development Concepts, Principles and Resource Materials," Rural Development Series, 1977 (Number 6). Aso, "Synthesis" and "A Selected Bibliography".

The Research Group, Inc., Techniques for Public Information, Participation Review and Comment: State Experiences and Suggested Approaches in Response to Title II of the Social Security Act. Atlanta: Research Group, Inc., 1976.

A. Tversky, "Choice by Elimination," Journal of Mathematical Psychology, 1972.

A. Tversky, "Elimination By Aspects: A Theory of Choice," Psychological Review, July, 1972.

A. Tversky & D. Kahneman, "Judgment under Uncertainty: Heuristics and Biases," Science, September 27, 1974.

U.S. Dept. of Transportation, Effective Participation in Transportation Planning, Vol. 1, Community Invovlement Processes, and Vol. II, A Catalog of Techniques. Washington, D.C.: Federal Highway Administration, U.S. Dept. of Transportation 1976.

J.N. Warfield, Societal Systems: Planning, Policy, and Complexity, Wiley Interscience, New York, 1976.

J.N. Warfield, "Improving Behavior in Policy Making," Approaches to Problem Solving, Number 2, The Academy for Contemporary Problems, Columbus, Ohio, 1975.

J.N. Warfield, "TOTOS: Improving Group Problem Solving" Approaches to Problem Solving, Number 3, The Academy for Contemporary Problems, Columbus, Ohio, 1975.

J.N. Warfield, et al, "Methods of Idea Management", Approaches to Problem Solving, Number 4, The Academy for Contemporary Problems, Columbus, Ohio, 1975.

BIBLIOGRAPHY
on
APPLICATIONS
of
INTERPRETIVE STRUCTURAL MODELING (ISM)

1. Baldwin, M. M. (Ed.), Portraits of Complexity, Battelle Monograph No. 9, Battelle Memorial Institute, Columbus, Ohio, 1975.

2. Braud, D. H., "Interpretive Structural Modeling in Louisiana", State of Louisiana Department of Urban and Community Affairs, Baton Rouge, August, 1977.

3. Braud, D. H., D. W. Irvin, and K. Kawamura, "Implementation of Interpretive Structural Modeling in a State Planning Context", Proc. 1976 Pittsburgh Conf. on Modeling and Simulation, Instrument Soc. of Am., Pittsburgh, 1976, 1152-1157.

4. Carss, B. W., L. D. Logan, and L. R. Miller, "Establishing Priorities for In-Service Programs Through Participatory Decision Making Using an Interpretive Structural Modeling Approach", presented at the Australian Council for Educational Administration Fourth National Conference, University of Queensland, Brisbane, Australia, Aug. 21-26, 1977.

5. Christakis, A. N. and W. J. Bogan, Jr., "The Concept of the Regional Environmental Learning System", Proceedings of the 1978 IEEE International Conf. on Cybern. and Society (Tokyo), IEEE, New York, 1978, 1001-1007.

6. El Mokadem, A. M., J. N. Warfield, D. M. Pollick, and K. Kawamura, "Modularization of Large Econometric Models: An Application of Interpretive Structural Modeling", Chapter 20 in Ref. 1 above.

7. Farris, D. R. and A. P. Sage, "On the Use of Interpretive Structural Modeling to Obtain Models for Worth Assessment", Chapter 17 in Ref. 1 above.

8. Fitz, R. W., "Interpretive Structural Modeling as Tool for Social Learning", Chap. 13 in Ref. 1 above.

9. Fitz, R. W., "Interpretive Structural Modeling and the Policy Planning Process", Proc. 1975 Pittsburgh Conf. on Modeling and Simulation, Instrument Soc. of Am., Pittsburgh, 1975, 773-777.

10. Fitz, R. W., D. M. Gier, and J. Troha, "A Methodology for Project Planning Using Interpretive Structural Modeling", 1977 Proc. of the International Conf. on Cybernetics and Society (Washington, D. C.), IEEE, New York, 1977, 297-302.

11. Fitz, R. W. and J. Troha, "Interpretive Structural Modeling and Urban Planning", same journal as in Ref. 10, 303-307.

12. Fitz, R. W., "Organizing for Development Planning: Methodology, Models, and Communication", 1976 Proc. of the International Conf. on Cybern. and Society (Washington, D. C.), IEEE, New York, 1976, 652-659.

13. Franklin, Clyde W., Jr., "The Intent of Sociology", Chapter 4 in Reference 1 above.

14. Geiger, D. and R. W. Fitz, "Structural Modeling and Normative Planning for Ecosystems", same journal as in Ref. 12, 660-666.

15. Hart, W. L. and D. W. Malone, "Goal Setting for a State Environmental Agency", Chapter 10 in Ref. 1 above.

16. Hawthorne, R. W. and A. P. Sage, "On Application of Interpretive Structural Modeling to Higher Education Program Planning", Socio-Economic Planning Sciences, Vol. 9, Pergamon Press, 1975, 31-43.

17. Hornbach, D., J. Calfee, and E. Zamierowski, "Anticipating the Consequences of Development Projects: An Example of a Water Utilization Project", same journal as in Ref. 12, 674-679.

18. House, Robert W., "Application of ISM in Brazil's Alcohol Fuel Program", IEEE SMC Transactions, July 1979, 376-381.

19. Jedlicka, A., "Interpretive Structural Modeling and Transfer of Technology to Subsistence Farmers in Mexico", presented at the 1977 American Institute for Decision Sciences meeting, Chicago, October 21, 1977.

20. Kawamura, K. and D. W. Malone, "Structuring Objectives in a Systematic Decision-Making Methodology", same journal as Ref. 9, 779-784.

21. Kawamura, K. and D. G. Sherrill, "Final Report on an Interpretive Structural Modeling Workshop" to the Eastern Energy and Land Use Group, The U. S. Fish and Wildlife Service, Battelle, Columbus, Ohio, July, 1977.

22. Malone, D. W., "Application of Interpretive Structural Modeling: Relating Factors for Urban Success", Chapter 16 in Ref. 1 above.

23. Malone, D. W., "An Introduction to the Application of Interpretive Structural Modeling", Chapter 14 in Ref. 1 above.

24. Malone, D. W., "Interpretive Structural Modeling: Overview and Status Report", same journal as Ref. 9, 767-772.

25. Malone, D. W., "Strategic Planning: Applications of ISM and Related Techniques", same journal as Ref. 5, 995-1000.

26. Mayhew, R. W., "A Systematic Approach in Management: Applications of Objective Tree Structure of Staff Development Planning", Chapter 5 in Ref. 1 above.

27. Mizoguchi, F., K. Tahara, and M. Saito, "Use of ISM to an Analysis of Expert Role in Simulation Modeling Process", same journal as Ref. 5, 983-988.

28. Nakayama, H., T. Tanino, K. Matsumoto, H. Matsuo, K. Inoue, and Y. Sawaragi, "A Methodology for Group Decision Support with an Application to Assessment of Residential Environment", same journal as Ref. 5, 643-648.

29. Ryoubu, M., S. Ikeda, and Y. Sawaragi, "A Prediction Model for Regional Economic System by Self-Organization Method", same journal as Ref. 5, 619-622.

30. Sage, A. P. and D. W. Rajala, "On the Use of Structure in Determination of Multi-Attribute Utility Functions", same journal as Ref. 3, 1199-1204.

31. Sugiyama, K., T. Hiramatsu, and Yasuo Shimazu, "Toward Development of Earthquake Disaster Relational Model", same journal as Ref. 5, 788-793.

32. Travis, Paul, "Commission Experiment a Lively, Frank Session", Dayton (Ohio) Daily News, Aug. 17, 1976, page 1.

33. Waller, R. J., "An Application of Interpretive Structural Modeling to Priority Setting in Urban Systems Management", Chapter 12 in Ref. 1

34. Waller, R. J., "Application of Interpretive Structural Modeling in Management of the Learning Disabled", Chapter 11 in Ref. 1.

35. Waller, R. J., "Interpretive Structural Modeling: Reflections on its Current State and Some Speculations about its Future Applications", same journal as Ref. 9, 785-789.

36. Warfield, J. N., "Intent Structures", IEEE Trans. on Syst., Man, and Cybern., March, 1973, 133-140.

37. Warfield, J. N., "Extending Interpretive Structural Modeling", same journal as Ref. 3, 1163-1167.

38. Warfield, J. N., "Constructing Operational Value Systems for Proposed Two-Unit Coalitions", Proc. 1973 Conf. on Decision and Control, IEEE, New York, 1973, 204-213.

39. Zamierowski, E., D. Hornbach, and R. Fitz, "Ecological Components of Climax Agriculture: An Example of Structuring Complex Feedback Systems", same journal as Ref. 12, 667-673.

ADDITIONAL APPLICATIONS

40. Crim, Karen O., "Use of Interpretive Structural Modeling in Environmental Studies at the Senior High Level, University of Dayton Report, Grant No. G007700611, UDR-TR-79-27, April, 1979.

41. Fitz, R. and J. Troha, "An Experiment in Mapping Urban Vitality: A Summary of the Dayton City Commission's Workshop on Interpretive Structural Modeling", Unpublished report, University of Dayton, Engineering Management Program, Nov. 15, 1976.

42. K. Kawamura and A. Christakis, "An Interpretive Structural Modeling (ISM) Planning Workshop", prepared for the United States Coast Guard, Report from Battelle Columbus Laboratories, May 25, 1979.

43. Warfield, J. N., "Systems Planning for Environmental Education", Proceedings of the International Conference on Cybernetics and Society, IEEE, New York, 1979.

BIBLIOGRAPHY
on
THE THEORY
of
INTERPRETIVE STRUCTURAL MODELING (ISM)

1. Fertig, J. and J. N. Warfield, "Relations and Decision Making" Proc. 1976 Pittsburgh Conf. on Modeling and Simulation, Instrument Soc. of America, Pittsburgh, 1976, 1177-1181.

2. Fitz, R. W. and D. Hornbach, "A Participative Methodology for Designing Dynamic Models through Structural Modeling", same journal as Ref. 1, 1168-1176.

3. Sage, A. P., Methodology for Large Scale Systems, McGraw-Hill, New York, 1977.

4. Sage, A. P., "On Interpretive Structural Models for Single Sink Digraph Trees", 1977 Proc. of the International Conf. on Cybernetics and Society, IEEE, New York, 1977, 308-314.

5. Warfield, J. N., "On Arranging Elements of a Hierarchy in Graphic Form, IEEE Trans. on Syst., Man, and Cybern., March, 1973, 121-132.

6. Warfield, J. N., An Assault on Complexity, Battelle Monograph Number 2, Battelle Memorial Institute, Columbus, 1973.

7. Warfield, J. N., "Binary Matrices in System Modeling", IEEE Trans. on Syst., Man, and Cybern., Sept., 1973, 441-448.

8. Warfield, J. N., "Developing Subsystem Matrices in Structural Modeling", IEEE Trans. on Syst., Man, and Cybern., Jan., 1974, 74-80.

9. Warfield, J. N., "Developing Interconnection Matrices in Structural Modeling", IEEE Trans. on Syst., Man, and Cybern., Jan., 1974, 81-87.

10. Warfield, J. N., Structuring Complex Systems, Battelle Monograph No. 4, Battelle Memorial Institute, Columbus, April, 1974.

11. Warfield, J. N., "An Interim Look at Applications of Interpretive Structural Modeling", Research Futures, 3rd Quarter, 1974.

12. Warfield, J. N., "Toward Interpretation of Complex Structural Models", IEEE Trans. on Syst., Man, and Cybern., Sept, 1974, 405-417.

13. Warfield, J. N., "Transitive Interconnection of Transitive Structures", Proc. 1975 Pittsburgh Conf. on Simulation and Modeling, Instrument Society of Am., Pittsburgh, 1975, 791-794.

14. Warfield, J. N., "Implication Structures for System Interconnection Matrices", IEEE Trans. on Syst., Man, and Cybern., Jan., 1976, 18-24.

15. Warfield, J. N., Societal Systems: Planning, Policy, and Complexity, John Wiley and Sons, Interscience Division, New York, 1976.

16. Warfield, J. N., "Crossing Theory and Hierarchy Mapping", IEEE Trans. on Syst., Man, and Cybern., July, 1977, 505-523.

17. Warfield, J. N., "The Interface Between Models and Policymakers", Journ. of Policy Analysis and Information Systems, January, 1979.

18. Warfield, J. N., "Some Principles of Knowledge Organization", IEEE Trans. on Syst., Man, and Cybern., June, 1979, 317-325.

19. Waller, Robert J., "Comparing and Combining Structural Models of Complex Systems", IEEE Trans. Syst., Man, and Cybern., Special Issue on Public Systems Methodology, September, 1979.

BIBLIOGRAPHY ON WORTH ASSESSMENT*

---

*FROM TECHNIQUES FOR WORTH ASSESSMENT BY NORWOOD P. KNEPPRETH, DAVID H. GUSTAFSON, AND RICHARD P. LEIFER OF THE UNIVERSITY OF WISCONSIN AND EDGAR M. JOHNSON OF THE ARMY RESEARCH INSTITUTE FOR THE BEHAVIORAL SCIENCES, ARMY RESEARCH INSTITUTE TECHNICAL PAPER 254, AUGUST 1974.

BIBLIOGRAPHY

Ackoff, R.L., S. Gupta and J. Minas. *Scientific method: Optimizing applied research decisions*. New York: Wiley, 1962.

Aumann, R. J. and J. B. Kruskal. Assigning quantitative values to qualitative factors in the naval electronics problem. *Naval Research Logistics Quarterly*, 1959, *6*, 1-16.

Bartlett, C., F. Heerman and S. Rettig. A comparison of six different scaling techniques. *Journal of Social Psychology*, 1960, *51*, 343-348.

Beach, B. Direct and indirect methods for measuring utility. Unpublished research report, University of Washington, Department of Psychology, July 1972.

Bechtel, G. Comparative scaling of unidimensional discrimination and similarity data. *Psychometrika*, 1966, *31*, 75-84.

Bechtel, G. Folded and unfolded scaling from preferential paired comparisons. *Journal of Mathematical Psychology*, 1968, *5*, 333-357.

Becker, G., M. DeGroot and J. Marschak. Measuring utility by a single-response sequential method. *Behavioral Science*, 1964, *9*, 226-232.

Bock, R. D. and L. V. Jones. *The measurement and prediction of judgment and choice*. San Francisco: Holden-Day, 1968.

Bush, J., M. Chen and D. Patrick. Cost effectiveness using a health status index: Analysis of the New York State phenylketonuria screening program. Unpublished report, University of California at San Diego, Department of Community Medicine, September 1972.

Churchman, C. W. and R. L. Ackoff. An approximate measure of value. *Operations Research,* 1954, *2*, 172-187.

Conrath, D. and F. Deci. The determination and scaling of a bivariate utility function. *Behavioral Science*, 1969, *14*, 316-327.

Coombs, C. H. Inconsistency of preferences: A test of unfolding theory. In W. Edwards and A. Tversky (Ed.), *Decision Making*. Baltimore: Penguin, 1967.

Coombs, C. H. and S. S. Komorita. Measuring utility of money through decisions. *American Journal of Psychology*, 1958, *71*, 383-389.

Cooper, L. G. Metric multidimensional scaling and the concept of preference. Los Angeles: Western Management Science Institute, Working Paper 163. October 1970.

Dalkey, N., B. Brown and S. Cochran. The Delphi method: Use of self ratings to improve group estimates. Rand Corp. Memo RM-6115-PR, November 1969. (AD 698 735)

Davidson, F. Utility and decision theory in a planning environment. Unpublished report, University of Virginia, McIntire School of Commerce, 1970.

Eckenrode, R. Weighting multiple criteria. Management Science, 1965, 12, 180-192.

Edwards, W. Probability-preference among bets with differing expected values. American Journal of Psychology, 1954, 67, 56-57.

Edwards, W. Utility, subjective probability, their interaction and variance preferences. Journal of Conflict Resolution, 1962, 6, 42-51.

Edwards, W. Social utilities. In Proceedings of the Symposium on Decision and Risk Analysis--Powerful New Tools for Management. Annapolis, Md.: U. S. Naval Academy. June 1971.

Ekman, G. Two generalized ratio scaling methods. Journal of Psychology, 1958, 45, 287-295.

Fischer, G. W. Multi-dimensional value assessment for decision making. University of Michigan, Engineering Psychology Laboratory Technical Report 037230-2-T. June 1972. (a)

Fischer, G. W. Four methods for assessing multi-attribute utilities: An experimental validation. University of Michigan, Engineering Psychology Laboratory Technical Report 037230-6-T, September 1972. (b)

Fishburn, P. Decision and value theory. New York: Wiley, 1964.

Fishburn, P. Independence, trade-offs and transformations in bivariate utility functions. Management Science, 1965, 11, 792-801.

Fishburn, P. Methods of estimating additive utilities. Management Science, 1967, 13, 435-453.

Fishburn, P. Utility theory. Management Science, 1968, 14, 335-378.

Fishburn, P. Utility theory for decision making. New York: Wiley, 1970.

Galanter, F. The direct measuement of utility and subjective probability. American Journal of Psychology, 1962, 75, 208-220.

Ginsberg, A. Decision analysis in clinical patient management with application to the pleural-effusion syndrome. Rand Corp. Memo R-751-RC/NLM, July 1971.

Goldberg, L. Five models of clinical judgment: An empirical comparison between linear and nonlinear representations of the human inference process. Organizational Behavior and Human Performance, 1971, 6, 458-479.

Guilford, J. P. Psychometric methods. New York: McGraw-Hill, 1954.

Gulliksen, H. Measurement of subjective values. Psychometrika, 1956, 21, 229-244.

Gustafson, D. H., I. Feller, K. Crane and D. Holloway. A decision theory approach to measuring severity in illness. Unpublished report, University of Wisconsin, 1971.

Gustafson, D. H., G. Kramer and G. Pai. A weighted aggregate approach to R&D project selection. AIIE Transactions, 1971, 3, 22-31.

Herstein, I. N. and J. Milnor. An axiomatic approach to measurable utility. Econometrica, 1953, 21, 291-297.

Hoepfl, R. and G. Huber. A study of self-explicated utility models. Behavioral Science, 1970, 15, 408-414.

Hoffman, J. and C. Peterson. A scoring rule to train probability assessors. University of Michigan, Engineering Psychology Laboratory Technical Report 037230-4-T, September 1972.

Huber, G. Multi-attribute utility models: A review of field and field-like studies. Management Science, 1974, in press.

Huber, G., R. Daneshgar and D. Ford. An empirical comparison of five utility models for predicting job preferences. Organizational Behavior and Human Performance, 1971, 6, 267-282.

Huber, G. and D. H. Gustafson. Some efforts to apply behavioral decision theory in the medical care field. Unpublished report, University of Wisconsin, September 1971.

Huber, G., V. Sahney and D. Ford. A study of subjective evaluation models. Behavioral Science, 1969, 14, 483-489.

Hurst, P. and S. Siegel. Prediction of decisions from a higher ordered metric scale of utility. Journal of Experimental Psychology, 1956, 52, 138-144.

Klahr, D. Decision making in a complex environment: The use of similarity judgments to predict preferences. Management Science, 1969, 15, 595-618.

Levine, J. M., J. B. Feallock, R. Sadacca and R. Andrews. Method for quantifying subjective costs of large numbers of image interpretation errors. ARI Technical Research Note 218. November 1969. (AD 704 706)

MacCrimmon, K. R. Decision making among multi-attribute alternatives: A survey and consolidated approach. Rand Corp. Memo RM-4823-ARPA, December 1968.

MacCrimmon, K. R. Improving the system design and evaluation process by the use of trade-off information: An application to northeast corridor transportation planning. Rand Corp. Memo RM-5877-DOT, April 1969. (PB 185 166)

MacCrimmon, K. R. and M. Toda. The experimental determination of indifference curves. Los Angeles: Western Management Science Institute, Working Paper 124, 1968.

Marquardt, R., J. Makens and H. Larzelere. Measuring the utility added by branding and grading. Journal of Marketing Research, 1965, 2, 45-50.

McKendry, J. M., P. C. Harrison, A. H. Birnbaum and R. Sadacca. Estimating the value of surveillance information using error cost matrices. ARI Technical Research Note 184. June 1967. (AD 667 390)

Miller, J. R., III. A systematic procedure for assessing the worth of complex alternatives. USAF ESD-TR-67-90, November 1967. (AD 662 001)

Miller, J. R. Assessing alternative transportation systems. Rand Corp. Report RM5865-DOT, April 1969. (PB 185 167)

Mosteller, F. and P. Nogee. An experimental measurement of utility. Journal of Political Economics, 1951, 59, 371-404.

O'Connor, M. F. The assessment of worth for multi-attributed alternatives: A survey. Unpublished report, University of Michigan, 1972.

Pai, G., D. Gustafson and G. Kiner. Comparison of three non-risk methods for determining a preference function for money. Unpublished report, University of Wisconsin, January 1971.

Pardee, F. S. and C. T. Phillips. The utility of utility theory in regional transportation mix analysis. Presented at the 36th National Meeting of the Operations Research Society of America, Miami, August 1970. (AD 711 497)

Pardee, F. S., C. T. Phillips and K. Smith. Measurement and evaluation of alternative regional transportation mixes: Vol II, Methodology. Rand Corp. Report RM-6324-DOT, August 1970.

Patrick, D., J. Bush and M. Chen. Comparing three methods for measuring social preference for levels of function. Unpublished report, University of California at San Diego, 1971.

Raiffa, H. Preferences for multi-attributed alternatives. Rand Corp. Memo RM-5868-DOT-RC, April 1968.

Royden, H., P. Suppes and K. Walsh. A model for the experimental measurement of the utility of gambling. *Behavioral Science*, 1959, *4*, 11-18.

Sellin, T. and M. E. Wolfgang. *The measurement of delinquency*. New York: Wiley, 1964.

Siegel, S. A method for obtaining an ordered metric scale. *Psychometrika*, 1959, *21*, 207-216.

Slovic, P. Consistency of choice among equally valued alternatives. *Proceedings of the 76th Annual Convention of the American Psychological Association*, 1968, *3*, 57-59.

Slovic, P. Analyzing the expert judge: A descriptive study of a stockbroker's decision process. *Journal of Applied Psychology*, 1969, *53*, 255-263.

Slovic, P. and S. Lichtenstein. Relative importance of probabilities and payoffs in risk taking. *Journal of Experimental Psychology Monograph*, 1968, *78* (No. 3, Part 2).

Stagner, R. Corporate decision making: An empirical study. *Journal of Applied Psychology*, 1969, *53*, 1-13.

Stauss, F. Comparison of five estimation techniques for subjective probability. Unpublished Master's thesis, University of Wisconsin, Department of Industrial Engineering, 1972.

Stevens, S. S. A metric for the social consensus. *Science*, 1966, *151*, 530-541.

Stevens, S. S. Issues in psychophysical measurement. *Psychological Review*, 1971, *78*, 426-450.

Stimson, D. H. Utility measurement in public health decision making. *Management Science*, 1969, *16*, 17-30.

Suchman, F. *Evaluative research*. New York: Russell Sage Foundation, 1967.

Summers, D. A., J. D. Taliaferro and D. J. Fletcher. Judgment policy and interpersonal learning. *Behavioral Science*, 1970, *15*, 514-521.

Suppes, P. and K. Walsh. A non-linear model for the experimental measurement of utility. *Behavioral Science*, 1959, *4*, 204-211.

Suppes, P. and M. Winet. An axiomatization of utility based on the notion of utility differences. *Management Science*, 1955, *1*, 259-270.

Swalm, R. O. Utility theory--insights into risk taking. *Harvard Business Review*, 1966, *44*, 123-136.

Thurstone, L. L. The measurement of values. Chicago: University of Chicago Press, 1959.

Toda, M. Indifference map method for estimating utility functions. Hokkaido Report of Psychology, HRP-1-71-12, August 1971.

Torgerson, W. S. Theory and methods of scaling. New York: Wiley, 1958.

Tversky, A. Utility theory and additivity analysis of risky choices. Journal of Experimental Psychology, 1967, 75, 27-36.

Von Neumann, J. and O. Morgenstern. Theory of games and economic behavior. New York: Wiley, 1944.

Von Winterfeldt, D. and W. Edwards. Costs and payoffs in perceptual research. In E. C. Carterette and M. P. Friedman (Eds.), Handbook of Perception. New York: Academic Press, in press.

Vroom, V. H. and E. Deci. The stability of post-decision dissonance: A follow-up study of the job attitudes of business school graduates. Organization Behavior and Human Performance, 1971, 6, 36-49.

Zaleska, M. and N. Kogan. Level of risk selected by individuals and groups when deciding for self and for others. Sociometry, 1971, 34, 198-213.

APPENDIX

COMPUTER IMPLEMENTATION OF ISM

PART ONE
and
PART TWO

by

Raymond L. Fitz
David R. Yingling
Joanne B. Troha
Karen O. Crim

University of Dayton

UDR-TR-79-79

Appendix to Volume 4:
Conducting Collective Inquiry
COMPUTER IMPLEMENTATION OF INTERPRETIVE STRUCTURAL MODELING

(Part One)

Written by:

Raymond L. Fitz, S.M.
David R. Yingling
Joanne B. Troha
Karen O. Crim

University of Dayton

September 1979

TABLE OF CONTENTS

Part One

Table of Contents iii

List of Figures. v

Introduction. 1

What is Interpretive Structural Modeling? 1

  Encouraging Participation in Collective Inquiry 2
  An Overview of an Interpretive Structural Modeling Exercise 3

Computer Equipment for Interpretive Structural Modeling 9

  Modular Approach to Equipment 9
  Module Descriptions 10
  Alternative Configurations 14

Strategies for Obtaining Equipment. 15

  Utilizing Community Resources 15
  Development of a Support Plan 16

Programmer's Overview of the Interpretive Structural Modeling Package . . . 17

  ISMS -- The System 18
  ISMS -- A User's View 20
  File Formats 22

Bibliography. 29

Attachment 1: Documentation of the ISMS Algorithms

Attachment 2: Interpretive Structural Modeling Software

Attachment 3: DELTA Charts for ISMS

Part Two

Attachment 4: FORTRAN Programs for ISMS

LIST OF FIGURES

Figure 1: ISM Notetaking Form 5

Figure 2: ISMS-UD Basic Block Diagram 19

Figure 3: ISMS-UD System Block Diagram 23

Figure 4: Sample Text File 24

Figure 5: Format of the Randomized Text File 26

Figure 6: Format of a Text Record within the Randomized Text File . 26

Figure 7: Format of Model Files 26

Figure 8: Suggested ISMS-UD Overlay Scheme. 27

COMPUTER IMPLEMENTATION OF INTERPRETIVE STRUCTURAL MODELING

Introduction

One of the chief obstacles to solving the environmental problems that confront the world today is the inability of both average citizens and policy decision-makers to deal with the complexity of these problems. Environmental problems, such as air and water pollution, are comprised of many factors and involve interrelationships that may be difficult to understand. As a result, solutions to environmental problems have often been too simplistic, and have not always improved the quality of the environment.

This appendix presents information about interpretive structural modeling, a method that can be useful in dealing with complex environmental issues in a collective inquiry setting. The first section discusses interpretive structural modeling as it can be used for collective inquiry and presents an overview of an interpretive structural modeling exercise. Later sections of the appendix discuss the computer equipment and the interpretive structural modeling (ISM) software package.

What is Interpretive Structural Modeling?

Interpretive structural modeling (ISM) is a computer-aided method to assist a group of people in studying and analyzing complex problems. It is appropriate to use ISM when the issue or problem under study can be broken down into the component parts that describe the situation. Participants in an ISM exercise define the structure of a complex system by focusing on the relationships between the elements of the system.

Some ISM applications include:

- a study of children with learning disabilities;
- work with neighborhood groups to identify factors involved in neighborhood crime;
- an analysis of the Goals for Dallas;
- a long-range planning study of the Sahel region of Africa;
- the identification of goals for a state-planning effort;
- defining the objectives of PLANALSUCAR, an agency involved in the Brazilian National Alcohol Fuels Program;
- establishing priorities for teacher in-service education programs; and
- a study of environmental issues by high school biology and social studies students.

Sometimes the model is developed to reflect the group's existing knowledge; at other times the model is structured after additional study about the issue. The results of an ISM exercise are:

- a greater understanding of the complex issue or problem through focused debate and clearly defined terms;
- an easily understood model or diagram that shows the structure of the issue; and
- some degree of order to the problem so that solutions can be more readily identified.

Therefore, the ISM method can be a useful tool for environmental education.

Encouraging Participation in Collective Inquiry

Many environmental educators are concerned with the question of how to encourage broader participation in collective inquiry. Increased citizen involvement in urban issues serves as a good example. Collective inquiry is more likely to lead to effective issue resolution when it:

1. takes into account the complexity of the problem or issue;
2. results from a wide base of influence and supporting resources; and
3. shows an awareness of the high interdependency that characterizes urban society.

When confronted with a complex environmental issue, people may be highly uncertain and ignorant about the dynamics involved. There is a need, therefore, for a learning process that can help people (citizens as well as political and organizational leaders) to create a shared understanding of the issue.

In Volume 3, Creating a Regional Environmental Learning System, we presented a discussion of the collective inquiry and action processes -- dialogue, decision, action, and evaluation -- and explained how collective inquiry can be used to address and resolve regional environmental issues. Interpretive structural modeling can contribute to more effective collective inquiry, by encouraging participation in the discussions about the environmental issue.

Some of the features of ISM that have appealed to collective inquiry groups make it an effective method for environmental education. These features include:

- provides a clearer understanding of the issue under study;
- makes maximum use of the group's previous research;
- organizes their collective knowledge;
- leads to consensus; and
- provides a way to communicate the results of the group's work.

An Overview of an Interpretive Structural Modeling Exercise

To do an ISM exercise, a group first decides on a meaningful set of elements and an appropriate relationship to study. The next step is to use the chosen relationship to discuss how the elements are related to one another. This discussion should lead the group to a new understanding of the total system of elements and relationships. The graphical representation, or structural model, which is produced with the aid of a computer, serves as the basis for developing this understanding.

During an ISM exercise the computer functions as a bookkeeper, displaying the questions for the group to discuss, maintaining logical consistency, and recording the responses. The computer is programmed to do this. This appendix includes detailed information about the computer programs for ISM.

During a typical ISM session the participants are seated at tables, with TV monitors located so that everyone can see one. A telephone receiver connects a small portable terminal with the computer. The TV monitors are wired to the terminal to provide a visual display for the participants. It is desirable to have a printer available to provide a typewritten copy of the session. A blackboard or flipchart is useful, and a special ISM Notetaking Form (Figure 1) should be provided for the participants.

An interpretive structural modeling exercise has six steps:

1. Orienting the exercise
2. Generating the elements and choosing the relationship
3. Structuring the elements with the relationship
4. Amending the model
5. Exploring the implications of the model
6. Documenting, evaluating, and communicating

The main purpose of the orientation is to establish clear expectations for the ISM exercise. This step includes such tasks as choosing an environmental topic to model, setting the schedule for the sessions, acquainting the participants with the ISM process, and clarifying the roles during the ISM exercise. The topic can be divided into specific areas for the participants to research individually or in task forces. Resource speakers can provide background information through lectures and additional resource materials.

In the second step of the ISM exercise, the group generates a list of elements that contribute to the environmental problem or issue and chooses the relationship that will be used for the modeling. During the discussions to produce this element list the participants can share their research on the environmental topic and develop an initial definition for each element.

NOTETAKING OF ISM SESSION

GROUP ______________________________ PAGE ________ OF ____________ PAGES

TOPIC ______________________________ DATE OF SESSION ____________________

Relationship	Y/N	Notes on discussion, definitions, voting, and other comments	Relationship	Y/N	Notes on discussion, definitions, voting, and other comments

Figure 1: ISM Notetaking Form

The third step of the ISM process involves the use of the computer to structure the elements with the relationship. For the structuring sessions, you will need access to the computer by way of a terminal, telephone connection, and television monitors. You will also need someone to set up the equipment and manage the terminal during the structuring sessions. In addition to the physical setup, you must also arrange to use the ISM computer software package. Detailed information about computer equipment and the ISMS-UD software package is presented in later sections of this appendix.

With the terminal, telephone connection, and television monitors in place, you are ready to begin structuring. During the structuring sessions, the computer prompts the group to systematically examine the relationships between the elements, by displaying questions on the monitors. As each question appears on the TV monitor, the participants discuss their views and arrive at a response. A yes or no vote is recorded at the end of the discussion for each question. The leader's role during structuring is to guide the discussion and call for a vote to answer the questions, and the terminal manager's role is to operate the computer equipment. At the end of the structuring, the computer prints out the information needed to draw the model that the group has produced.

Experience has shown that it always takes longer to model than a beginner might expect. The time required to produce a model is a function of the number of questions that the group must answer, and the number of questions is a function of the number of elements to be modeled. As the number of elements increases, the number of questions to be answered rises exponentially. The computer is programmed to infer some of the relationships, thereby reducing the number of questions to be answered. In addition, when a new element is introduced the computer calculates the best sequence of questions to ask so that the group can examine all the interconnections in the most efficient way.

Notetaking is also an important part of the modeling sessions. Notes on each relationship can be helpful in later review of the model, and may point to areas that require further study. A special form (Figure 1) for notetaking during an ISM session is available.

At certain points, you may stop the sequence of questions and use the computer to display the results of the modeling, from which you may draw a diagram of the model. The computer does not actually draw the model, but it prints out the information you need to make the drawing. The group can examine the drawing and review its understanding of the issue.

Sometimes the model includes a cycle among several elements. Within a cycle, every element is related to every other element. When a model includes a cycle, the group can give structure to that part of the model by discussing the strength of relationships between the cycle elements. By weighting the elements, it is possible to identify the strongest relationships within the cycle, thus further clarifying the model's structure.

In interpretive structural modeling, the final model emerges slowly, as much as a result of amending as of the initial structuring process. Often amending will be done both during modeling sessions and through outside staff work. Periodically, the group should examine the structural model for internal consistency and logic, and then amend as necessary. The first model may not be satisfactory -- it may contain relationships that do not exist, or perhaps an absence of relationships that should be there. The ISM software includes the capability to make any refinements in the model that the group feels necessary.

Sometimes the easiest way to amend the model is to remove one of the elements from the model and rework it through the usual series of questions. In other words, prompted by the computer, the group reworks the element back into the model. The amending process calls for a word of warning: the leader should be careful that the model the group has spent several hours to construct is not destroyed by haphazard amending.

To achieve a useful model the group should take time to explore the implications of the model. This might include, for example, discussing the assumptions that are implicit in the model. Another way to approach the model would be to identify key points (for example, points where the model branches). If these points can be considered strategic places to intervene, the group could spend some time discussing what persons or institutions might be most effective in resolving the issue under study.

Consider, too, how the model can be helpful in evaluating possible outcomes of alternative actions. Does the model suggest anything about what is right or wrong with our present approaches to the environmental issue or problem?

To summarize these possible discussion points, we offer this list of suggested questions:

1. How can the model be interpreted?
2. What was learned?
3. What can be said about the assumptions behind the model?
4. Does the model suggest any course of action that could help resolve the environmental issue?
5. Who should be involved in the solution?
6. Other ideas?

The final step in an ISM exercise is to document the process and the model for later reference, although the extent to which a model is formally documented will depend on how you intend to use it. The documentation might include a summary of how the model was developed, comments from the discussion, the element list and definitions, conclusions drawn from the model, and so on. Usually the documentation of the ISM exercise is only adequate as a summary for the participants. The group must use other methods to communicate the results to others who did not participate in the ISM sessions.

Since ISM exercises vary considerably (leaders, participants, issues, time spent), each exercise should be evaluated in terms of the purposes and goals of that particular exercise. Avoid comparisons with other ISM exercises, using comparison only to assist planners and leaders in improving their skills with the ISM process.

It is clear that the steps in an interpretive structural modeling exercise closely correspond with the processes of collective inquiry and action discussed in Volume 3. Field testing, described in Volume 4, Conducting Collective Inquiry, and previous applications of ISM have shown it to be an effective method for collective inquiry.

Subsequent sections of this appendix present information about the computer equipment and the ISM software package needed to use interpretive structural modeling for collective inquiry and action.

Computer Equipment for Interpretive Structural Modeling

This section provides a description of computer equipment that is suitable for implementing ISM. First, there is a discussion of the modular approach to the computer equipment. The modular approach allows freedom to select equipment that meets budget and computer capability requirements. Next, we discuss alternative equipment for each module. Finally, the alternative equipment is combined to provide some alternative system configurations. Most of these system configurations have already been used and proven in during interpretive structural modeling exercises.

The equipment and the systems recommended in this section are representative of the current technology available. The reader should be aware of the fact that prices fluctuate with the state of the art in computers and computer peripherals. When it comes time to acquire equipment, be sure to check various models and different manufacturers' equipment to determine how to get the most computer capability for your dollar.

Modular Approach to Equipment

In this section, we discuss a modular approach to configuring computer equipment for ISM. A modular approach facilitates the design of a system to maximize capability and minimize cost. We use this approach because of the varying budgets of possible

80

purchasers. The computer equipment for ISM has four distinct subsystems. These subsystems are: the computer system, display units, graphics units, and voting systems. The first two subsystems are required; the last two are optional.

The computer system is the heart of the equipment. It consists of the central processing unit, memory, and peripherals. The computer is the device that executes software and drives the display and graphics units. We will describe the alternative computer systems later in this section.

The display units (more commonly called terminals) provide the interface between the users and the computer equipment. A display unit looks similar to a typewriter keyboard, but it usually has a CRT (cathode ray tube), i.e., TV-type screen or monitor, and/or a printer mechanism. The CRT-type units usually have a plug available so that a limited number of extra monitors can be attached, which increases the number of people able to see the computer's output. You will need at least one display unit.

A graphic unit is also a display unit, but it is considered separately because of the additional functions it can perform. The graphic units (graphics terminals) look almost identical to the display units; both have keyboards and CRTs. However, the graphic unit adds the capability of color, bar graphs, and other features that a display unit does not have. Since it costs ten to fifteen times more than a display unit, a graphic unit is an optional device for ISM.

The fourth subsystem is the voting unit. A voting unit is electronic equipment that can accept vote "values" (weighted voting) and display the results. These units allow users to vote secretly (instead of by a hand vote) and provide many statistics related to the vote. This device allows the rapid and accurate tally of votes when large numbers of people are involved. Due to its high cost, the voting unit is also optional.

Module Descriptions

In this section we describe alternative modules for each of the four subsystems. To facilitate the purchase and maintenance of equipment, this selection is limited to off-the-shelf units. There are many equipment manufacturers; the selection of the following equipment is based upon manufacturer's reputation and warranties, as well as our experiences using the equipment.

Computer Systems

One computer system that could be used for interpretive structural modeling is that of a university, local corporation, or timesharing computer company. This would involve contracting with the computer installation an access agreement that allows the use of the computer resources. There are some major economic advantages to this arrangement. First, the purchase and maintenance costs of a computer system are completely eliminated. Second, since the computer system will be accessed via the community telephone system, the resources of this large computer system could be available to every phone subscriber. Some disadvantages are that the use of this system could be restricted to certain times of the day and certain days of the week. Also, computer malfunctions can cause disruptions of computer sessions. The cost of the using the system can also be limiting. Ideally, timesharing costs should be based on a cost per access. However, some computer services also include a minimum monthly charge that could exceed the amount of time used. Since cost of a timesharing system such as this varies, no price estimate can be given.

The second computer system is the TRS-80, which falls in the $1,000-$4,000 price category, as of mid-1979. It is a computer system manufactured by Tandy Electronics and marketed by Radio Shack. The TRS-80 is easily maintained, since there are over 6,000 Radio Shack stores throughout the United States and in nine foreign countries. Expanding and upgrading the system are easy to do with the peripherals manufactured by Tandy and numerous small companies. A disadvantage of this system is that the expansion is limited to a single user. The execution of very large programs is limited to the amount of memory in the system.

The Cromemco System Three Computer has been selected as the third computer system. In mid-1979, its price is approximately $10,000 to $20,000. The System Three Computer is a table-top system which will provide fast execution of RELS tasks, as well as other tasks such as word processing, inventory, and payroll. The system is easily expandable, utilizing the industry standard S-100 bus architecture, which makes available low cost memory and peripheral interfaces. Up to eight simultaneous users are able to use the system, and no special environmentally controlled rooms are required. The one major disadvantage with the system is that it will require a maintenance contract from an outside company.

The last computer system to be discussed is the Hewlett-Packard 3000 Series II computer system. This is a full scale computer system costing over $140,000. It is capable of handling the processing requirements of local governments and school systems. The computer can be easily expanded (although not at a low cost) to provide support for up to 53 simultaneous users. Many upgrade and expansion options are available; they permit the system to be tailored to specific needs. The large initial investment and maintenance contract prices, however, might be prohibitive. A system of this size requires a staff of personnel to operate it and to provide programming help. It should be noted that a computer system of this size is likely to be more than is needed for collective inquiry in a RELS.

Display Units

The first display unit is the Digi-Log Model Telecomputer II CRT terminal. This terminal features a built-in acoustic telephone coupler, a slave port for a printer, 40 and 80 characters per line, and a video output jack for connecting compatible video equipment, such as extra monitors and video recorders. This device costs about $1,500, and requires a closed circuit television (CCTV) monitor (about $150) for operation.

The second display unit is the Teleray 1000 series intelligent CRT terminal. It features user programmable functions; software selectable character size, selectable fields including dim, inverse video, blinking, and underline; and a slave port for a printer. It requires an external acoustic coupler (about $200), if a telephone interface is to be used. Currently, the Teleray 1000 costs about $900.

Both types of display units are capable of driving additional video monitors. The video monitors are of the type used in a CCTV system with a composite video input. It should be noted that a large screen television can also be used as a video monitor. Any of the Advent model (or equivalent) large screen displays are compatible with the above units. A large screen display costs over $4,000. However, many metropolitan areas, there are audio/video specialty shops that will rent a large screen display for about $150 a day.

The third display unit, the Texas Instruments Model 745 Silent 700, which is a printer type device. This unit weighs 15 pounds and has its own built-in acoustic telephone coupler. It is a complete unit in itself. An optional $50 interface allows this unit to double as an RS-232-C printer for the above two display units. With a price of $1,500, this dual-purpose unit is an excellent choice.

The fourth display unit, the Texas Instruments Model 800 terminal is also a printer type device. This terminal features a printer speed of 120 characters per second and an upper and lower case character set. It can also be used as a hardcopy printer for an RS-232-C compatible device. Its price is approximately $2,300.

Graphics Units

The recommended graphics units are the Tektronix 4027 and the Ramtek Micrographics color graphics terminals. These terminals feature color displays, slave printer and plotter ports, user definable fonts, and the capability of driving additional CCTV monitors. Devices such as the ones named above cost over $10,000.

Voting Devices

At this writing, only one commercially manufactured voting device, called Consensor, is available. This unit will support up to 16 voters. It visually displays statistics such as the distribution, average weighting, and the mean of the vote distribution. Its display device displays the information either numerically or in a histogram format. The price of this unit is about $8,200 -- plus installation.

Alternative Configurations

In this section we suggest some alternative configurations of the above mentioned equipment. Although not all configurations will be optimum for every situation, they should serve to provide a direction for the eventual equipment selection. Not included in these configurations are the cables and the extension cords required. These items are of a variable cost and can be supplied by the equipment manufacturers.

Configuration 1 is designed around the use of a timesharing computer system. The equipment recommended is the Digi-Log CRT terminal, Texas Instruments Model 745 display unit, and six CCTV monitors. This equipment configuration allows computer sessions with up to twelve participants. By adding a large screen display, approximately fifteen to thirty participants can be accommodated. The approximate total cost is $4,500, excluding the large screen display.

Configuration 2 is also designed around the use of a timesharing computer. It consists of equipment in configuration 1, with the addition of the Tektronix 4027 color graphics terminal. This addition costs another $10,000 and gives color display capability.

Configuration 3 is the Radio Shack TRS-80 microcomputer system, which is recommended for single use situations. This system provides sufficient computing power for the ISM software.

Configuration 4 consists of the Cromemco System Three computer, and any combination of display and graphics units. This configuration will provide support for the ISM software as well as for word processing, inventory, and payroll with up to eight simultaneous users. The cost of the basic system is about $10,000 for one user; additional user capability costs approximately $2,000 each.

Configuration 5 is a full-sized computer system that can support up to 54 users and provide such services as on-line record keeping, payroll, and others. It consists of the

Hewlett-Packard 3000 series minicomputer and any combination of display and graphics units. Such a system would cost over $160,000.

Manufacturers

Cromemco, Inc.
280 Bernado Avenue
Mountain View, CA 94040

Teleray
Division of Research, Inc.
Box 24064
Minneapolis, MN

Hewlett-Packard Co.
Computer Systems Groups
100 Wolfe Road
Cupetino, CA 95014

Radio Shack
Division of Tandy Corporation
2617 W. 7th Street
Fort Worth, TX 76107

DIGI-LOG Systems, Inc.
Babylon Road
Horsham, PA 19044

Texas Instruments
Digital Systems Division
P.O. Box 1444
Houston, TX 55424

Consensor
Applied Futures, Inc.
22 Greenwich Plaza
Greenwich, CT 06830

Strategies for Obtaining Equipment

This section presents strategies for obtaining the computer equipment needed for interpretive structural modeling. We discuss some strategies for acquiring computer equipment and recruiting personnel capable of interconnecting and operating the equipment. The reader can apply these and other appropriate strategies to configure and support a computer system suitable for interpretive structural modeling.

Utilizing Community Resources

One community resource to tap is the computer professional -- someone whose occupation is to program, design, and procure computer systems. Computer professionals may be vital to the successful implementation of ISM. Ideally, you will form a team of computer professionals with varying expertise. Systems analysts, computer engineers, and programmers all have skills to contribute. A systems analyst is concerned with the computer as a total system to serve a useful function, such as banking, payroll, and record systems. Since this total system incorporates both the hardware (equipment) and the software (programs), a systems analyst has broad knowledge that should be valuable. The computer engineer has a college degree in either electrical or computer engineering. The engineer's job is to design the computer equipment or to configure computer equipment to perform tasks related to production, energy management, etc. Although the computer engineer is basically hardware oriented, his or her expertise is a valuable asset, especially when utilizing borrowed and donated computer equipment. The programmer's job is to program computers to perform a variety of tasks. The programmer can help maintain the software and install it on your computer system. When assembling a team of computer professionals, try to recruit at least one of each type. These people work for such institutions as banks, schools, universities , and businesses.

The second community resource to investigate is the wealth of computer equipment. Possibly, computer equipment can be begged, borrowed, or bought from universities, local companies, and local, state, and Federal agencies. A moderately large local university probably has a timesharing computer system and a meeting room equipped with the required electronic devices. Local companies might donate their old computer equipment. The donation of computer equipment by state and Federal agencies is another way to obtain equipment. A RELS is likely to qualify for a discount on the purchase of new and used computer equipment from manufacturers and local computer stores.

Probably the best way to locate sources for the use, donation, and purchase of computer equipment is to rely on the computer experts. They will know of local computer installations, equipment in their company (or another) that is slated for replacement, and local computer vendors.

Development of a Support Plan

Because of the large investment of time and money, some support plan for the computer system is necessary. Computer systems require support for both the system hardware and the software. The support or maintenance of the hardware is taken care of during the warranty period; after that period, support is provided through a maintenance service contract or by a computer engineer or technician. The system software requires maintenance whenever errors in the programs are found. No matter how many times a computer program has been used, some latent programming errors may still exist -- especially within a large, complex program such as the interpretive structural modeling software package. The computer programmer, or another computer professional on your team, can correct these errors.

In summary, the details of the support plan depend on the specific equipment configurations you are using. In general, we recommend you develop a support plan early, allowing for as many contingencies as possible, so you can maximize the use of your computer equipment and software packages.

Programmer's Overview of the Interpretive Structural Modeling Software Package

The purpose of this programmer's overview is to document the Interpretive Structural Modeling Software (ISMS) package developed by the University of Dayton. ISMS-UD version 2 is a software package consisting of three interactive FORTRAN IV programs that aid in conducting ISM sessions. We document these programs in such a way that a computer professional can easily install them on other existing computer systems. Earlier versions of ISMS-UD have been successfully installed on various large and small computer systems, from small PDP-11's to CDC Cyber's. Included at the end of this section is a partial list of facilities that have installed ISMS-UD.

This section proceeds as follows. First, we offer a general description of ISMS-UD to acquaint the reader with the system scenario. Second, we discuss the installation of ISMS-UD from a user's point of view. Then we cover the installation details, including file formats and core storage requirements. Four attachments supplement this documentation. Attachment 1 is the detailed documentation of the machine algorithms that comprise ISMS-UD. Attachment 2 is the interpretive structural modeling software. It is a user's manual for ISMS-UD, as it has been installed at the University of Dayton. DELTA charts for ISMS-UD are included as Atttachment 3. Attachment 4, which is bound separately from this part of the appendix, consists of the FORTRAN programs for ISMS-UD.

ISMS -- The System

Interpretive structural modeling is a method that helps people think and communicate more effectively about complex issues and problems. ISMS is a series of algorithms that have been coded to allow machine execution of ISM. Warfield (1976) has presented previous discussions of these algorithms. Warfield's book will serve as an excellent reference for the documentation of the ISM algorithms.

ISMS-UD consists of three interactive FORTRAN IV programs -- ISMS-UD, CYCLE, and MAKEIT. The ISMS-UD program includes the rudiments of ISM; that is, embedding (Transitive Bordering Method) and some sophisticated digraph amending algorithms. CYCLE permits the resolution of cycles that may be included in an ISM digraph output of the ISMS-UD program. Resolution thresholds, geodetic cycles, and their associated paths are the primary outputs of CYCLE. The third program in the package, MAKEIT, allows both ISMS-UD and CYCLE to present queries to the user in an English text format. MAKEIT accepts a sequentially oriented file, usually created by the host computer text editor, and reformats it into another file that is randomly accessed by ISMS-UD and CYCLE. Figure 2 is a block diagram of the system.

ISMS-UD was designed and developed on the University of Dayton Univac Series 70/7 timesharing computer system under the VS/9 version 3.5 operating system and FORTRAN IV compiler BGFOR. The Univac system is an updated RCA Spectra, which is a hardware duplicate of an IBM 360 but with virtual storage facilities. Although virtual storage is not required for the installation of ISMS-UD, it is desirable.

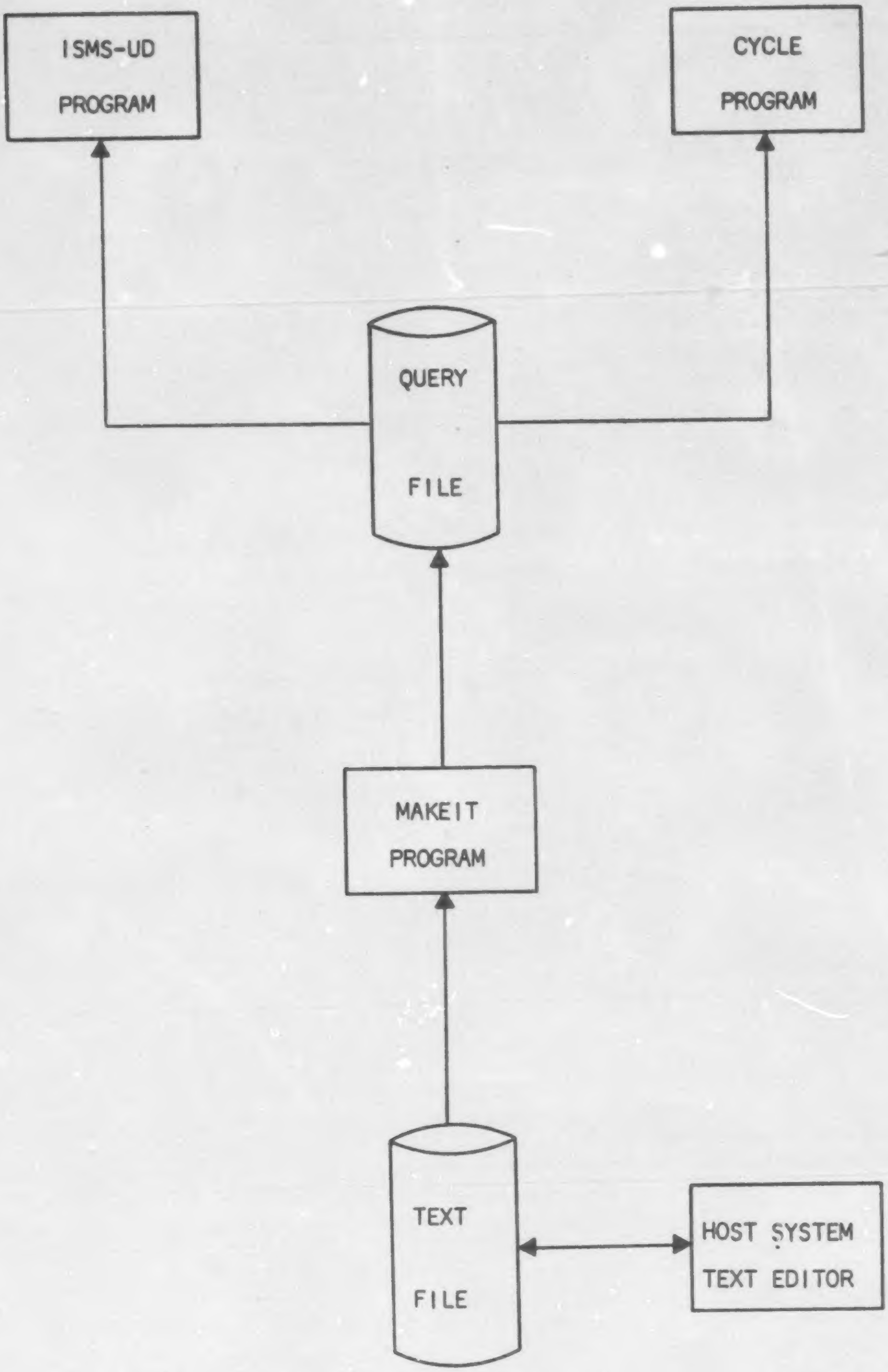

ISMS-UD PROGRAM
CYCLE PROGRAM
QUERY FILE
MAKEIT PROGRAM
TEXT FILE
HOST SYSTEM TEXT EDITOR

Figure 2: ISMS-UD Basic Block Diagram

ISMS -- A User's View

This section provides the installer with information needed by the user of ISMS-UD. Since the operating systems/executives on various computers differ, we present a generalized discussion of installation alternatives. The University of Dayton ISMS-UD user's guide (Attachment 2) supplements this section. The user's guide serves both as documentation of the software and as a model for user's guides for other installations.

The potential users of ISMS range from teachers in the classroom to researchers in laboratories to public administrators in the meeting room. Due to this wide diversity of users, it is important to install ISMS in such a way that little or no knowledge of the host computer job control language (JCL) is required. There are two approaches to this. One way (and probably the best) is to incorporate the operating system interface as part of the ISMS package. The software would query the user for names of disk files and check for validity of files, etc. This technique usually requires the writing of machine language routines to gain access to file handling routines via FORTRAN language. The second way is to incorporate JCL into "submit" or "catalogued procedures" (hereafter referred to as PROC files), if the operating system used supports this feature.

Following are the three PROC files that were written for the execution of the ISMS-UD software on the University of Dayton Univac Series 70 timesharing computer. These files provide disk file linkage and finally execute the respective program. The "&" (ampersand) allows parameters specified by the user to become part of the JCL.

```
/PROCEDURE N, (&SEQTEXT)
/FILE &SEQTEXT, LINK=DSET10, FCBTYPE=SAM
/FILE RND.&SEQTEXT, LINK=DSET08, FCBTYPE=ISAM, SPACE=(3,3)
/EXECUTE MAKEIT.E
/RELEASE DSET10
/RELEASE DSET08
/ENDPROC RETURN=(PRIMARY)
```

The above PROC is named "MAKEIT" and is invoked by:

```
/DO MAKEIT, (ENV.TEXT.1)
```

where ENV.TEXT.1 is the name of a user-created text file.

The PROC file first links ENV.TEXT.1 to FORTRAN unit reference 10. The second operation is to create a file named "RND.ENV.TEXT.1" as an output file to hold the randomized text. Next, the interactive program MAKEIT.E is invoked and the user then interacts with it. Upon termination, the FORTRAN logical units are released and the PROC is ended.

The most important thing to note about this PROC is the creation of an "invisible" (invisible to the user) file to hold the randomized text. The user only has to remember the name of his or her original text file to access the second "invisible" file. The next two PROC files use the "invisible" file.

```
/PROCEDURE N,(&MODEL, &TEXT)
/FILE &MODEL, LINK=DSET10, FCBTYPE=ISAM
/FILE RND.&TEXT, LINK=DSET08, FCBTYPE=ISAM
/EXECUTE ISMS-UD.E
/RELEASE DSET10
/RELEASE DSET08
/ENDPROC RETURN=(PRIMARY)
```

The above PROC is named "ISMS-UD" and invoked by:

```
/DO ISMS-UD, (ENV1.MOD, ENV.TEXT.1)
```

where "ENV1.MOD" is the file where the model information is to be stored and "ENV.TEXT.1" is the same file from above. It should be noted that the file "RND.ENV. TEXT.1" is actually linked.

```
/PROCEDURE N, (&CYCLE, &TEXT)
/FILE &CYCLE, LINK=DSET20, FCBTYPE=ISAM
/FILE RND.&TEXT, LINK=DSET08, FCBTYPE=ISAM
/EXECUTE CYCLE.E
/RELEASE DSET20
/RELEASE DSET08
/ENPROC RETURN=(PRIMARY)
```

The above PROC is named "CYCLE" and invoked by:

```
/DO CYCLE, (ENV1.CYC, ENV.TEXT.1)
```

where "ENV1.CYC" is the file where the cycle information is to be stored and ENV.TEXT.1 is the same file from above.

The greatest flexibility of ISMS-UD is its ability to access data files. The user of ISMS is primarily concerned with the models he or she has stored on disk files. For this reason, ISMS-UD should be installed on a computer system that is file oriented. This point cannot be stressed strongly enough.

Here we might note the design philosophy behind ISMS-UD. There were many earlier versions of ISMS-UD -- each of which had different designs and characteristics. ISMS-UD Version 2 incorporates what we felt to be the best operating characteristics for software of this type. ISMS-UD will not tutor the user in its use. It is imperative that the user have some knowledge of ISM and its basic principles. The software does have error traps so that the user can never input any erroneous data. The user will gain confidence in ISMS after a small amount of practice on the terminal.

File Formats

ISMS-UD's system block diagram is shown in Figure 3. ISMS-UD is an interactive system, which means that the user must respond to queries for input from the programs while they are executing. Examples of such user-provided input might be the answering of queries that specify a binary matrix representing a hierarchy, operations to be performed on this matrix, results to be displayed, and the like. For the user's convenience, the results of each matrix operation are automatically stored on the user-specified data file, allowing the programs to be terminated (normally or due to computer hardware errors) at any stage and restarted at a later time without loss of the matrix realization. There are three types of files that ISMS-UD will accept. They are: 1) sequential text files, 2) random access text files, and 3) model files. Each of the three types will be discussed below.

The sequential text file is used as input to the program MAKEIT and contains control and text records. Figure 4 shows an example of a sequential text file. The control records precede the text desired for the four types of text records -- R1 clause, R2 clause, R3 clause, and element text. The control records are identified by a "/" (slash)

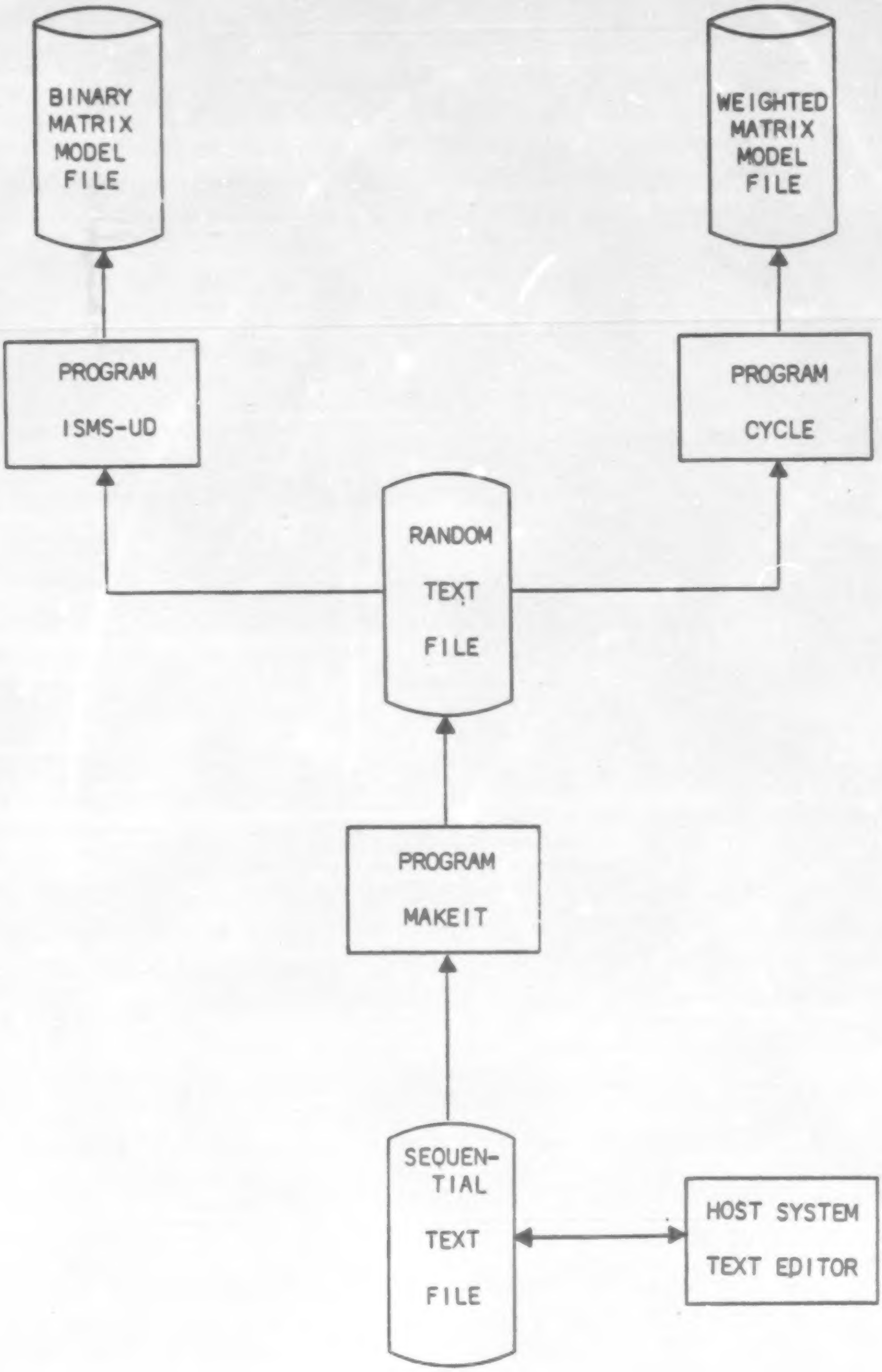

BINARY MATRIX MODEL FILE
WEIGHTED MATRIX MODEL FILE
PROGRAM ISMS-UD
PROGRAM CYCLE
RANDOM TEXT FILE
PROGRAM MAKEIT
SEQUENTIAL TEXT FILE
HOST SYSTEM TEXT EDITOR

Figure 3: ISMS-UD System Block Diagram

in the first column; therefore, no text record should begin with a "/". All of the four types of text records may have up to 10 lines with a maximum of 60 characters on each line (600 characters). The sequential text file, as its name implies, is accessed sequentially by MAKEIT. The sequential text file may be created and maintained by the host computer file editor, or it could simply be a deck of computer cards.

```
/R1
DOES
/R2
AGGRAVATE OR INTENSIFY
/R3
IN MOST CASES?
/EL
CITIZEN INSECURITY IN THE ABC NEIGHBORHOOD
/EL
GOOD MARKET FOR STOLEN GOODS
/EL
INADEQUATE CRIMINAL REHABILITATION
/EL
LACK OF SUMMER JOBS FOR THE YOUTH IN THE ABC AREA
/EL
LACK OF SEVERITY OF PUNISHMENT FOR CRIMES
/EL
LOW VISIBILITY OF UNIFORMED POLICE
//END OF ELEMENT LIST
```

Figure 4: Sample Text File

The random text file is required by programs ISMS-UD and CYCLE, if the user opts to have these programs present English text queries instead of the standard numeric type. The ISM method requires that elements be accessible in random order; that is, although the elements are ordered by the user in some desired sequential order when creating the sequential text file, the introduction of user responses to relational queries determines a new nonsequential presentation of the future queries. For this reason, a random (direct) access file is utilized. The random text file, created by MAKEIT, contains all of the textual information of the sequential file, but has the property that the text of any specific element may be retrieved independently of the positioning of the file due to a previous access. The format of the randomized text file is shown in Figure 5. Figure 6 shows the format of an individual record of the random text file.

There are two types of model files -- binary and weighted. The binary model file is used by the ISMS-UD program and contains the binary reachability matrix associated with the ISM method. The weighted model file is used by the CYCLE program and contains a weighted adjacency matrix. These two files are accessed sequentially and are opened and closed a number of times during program execution. The formats of both the binary and weighted matrix files are shown in Figure 7.

The ISMS-UD software has been designed in such a way that the binary and weighted operation matrices are core resident in order to speed up execution time. Therefore, the size of these matrix data structures directly affects the amount of core storage required due to the static allocation of memory characteristics of FORTRAN IV. On a computer with a limited amount of interactive core storage, ISMS-UD might be implemented with element and cycle resolution limits of something less than the current limit of 128 and 50. The ISMS-UD program requires 150K bytes, CYCLE requires 120K bytes, and MAKEIT requires 48K bytes on the Univac Series 70/7 computer. These core requirements should be somewhat lower on other computer systems -- the Univac system used at the University of Dayton requires high overhead for FORTRAN-run time routines. The overlay structures used for program ISMS-UD and CYCLE are shown in Figure 8.

The binary matrices used in the ISMS-UD program are of the type "LOGICAL *1". This defines the FORTRAN logical mode and allocates only one byte per matrix cell. Note that this data structure requires one fourth the amount of core storage as a "LOGICAL" matrix. The programmer should try to use the data structure in his or her computer that uses the smallest amount of directly addressable storage. The use of "IMPLICIT INTEGER *2" statements utilizes halfword precision for all integer variables, thereby further reducing the core requirements. All variables requiring full word storage have been declared as "INTEGER".

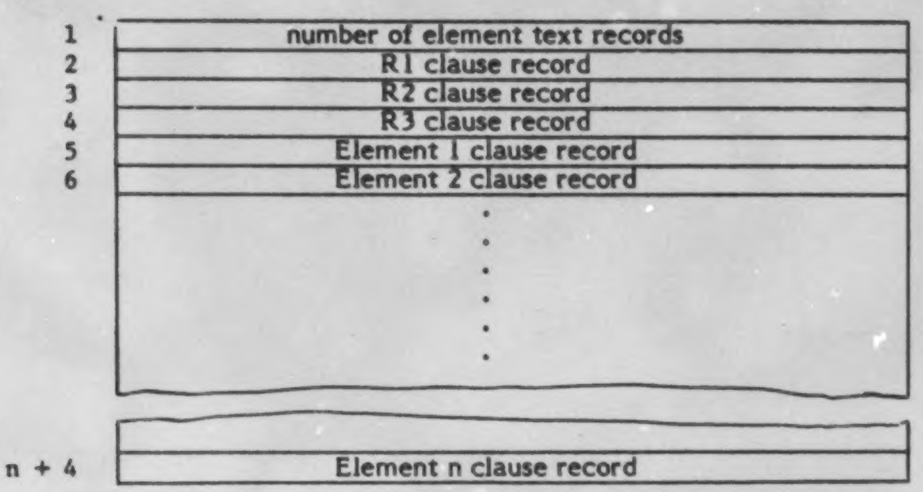

1
number of element text records
2
R1 clause record
3
R2 clause record
4
R3 clause record
5
Element 1 clause record
6
Element 2 clause record
n + 4
Element n clause record

Figure 5: Format of the Randomized Text File

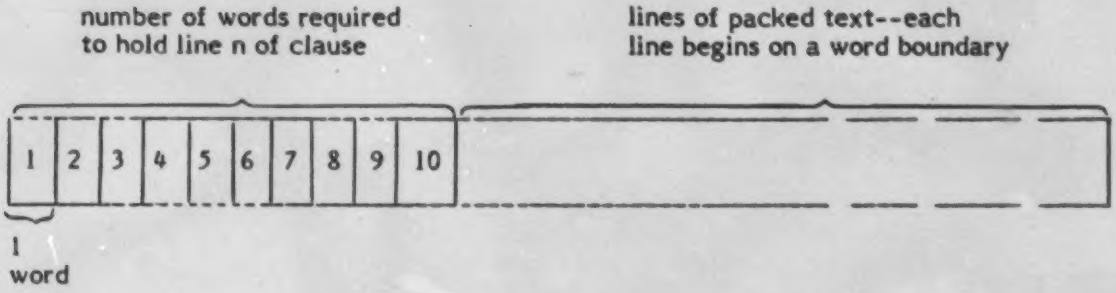

number of words required
to hold line n of clause
lines of packed text--each
line begins on a word boundary
1
2
3
4
5
6
7
8
9
10
1
word

Figure 6: Format of a Text Record within the Randomized Text File

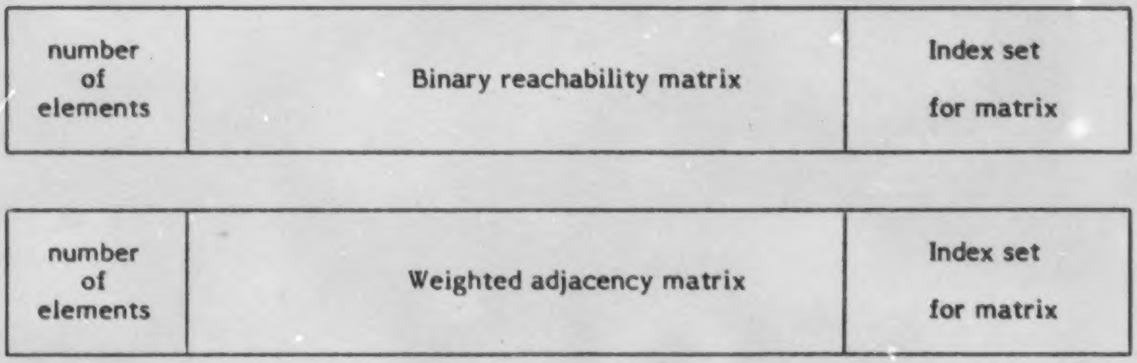

number
of
elements
Binary reachability matrix
Index set
for matrix
number
of
elements
Weighted adjacency matrix
Index set
for matrix

Figure 7: Format of Model Files

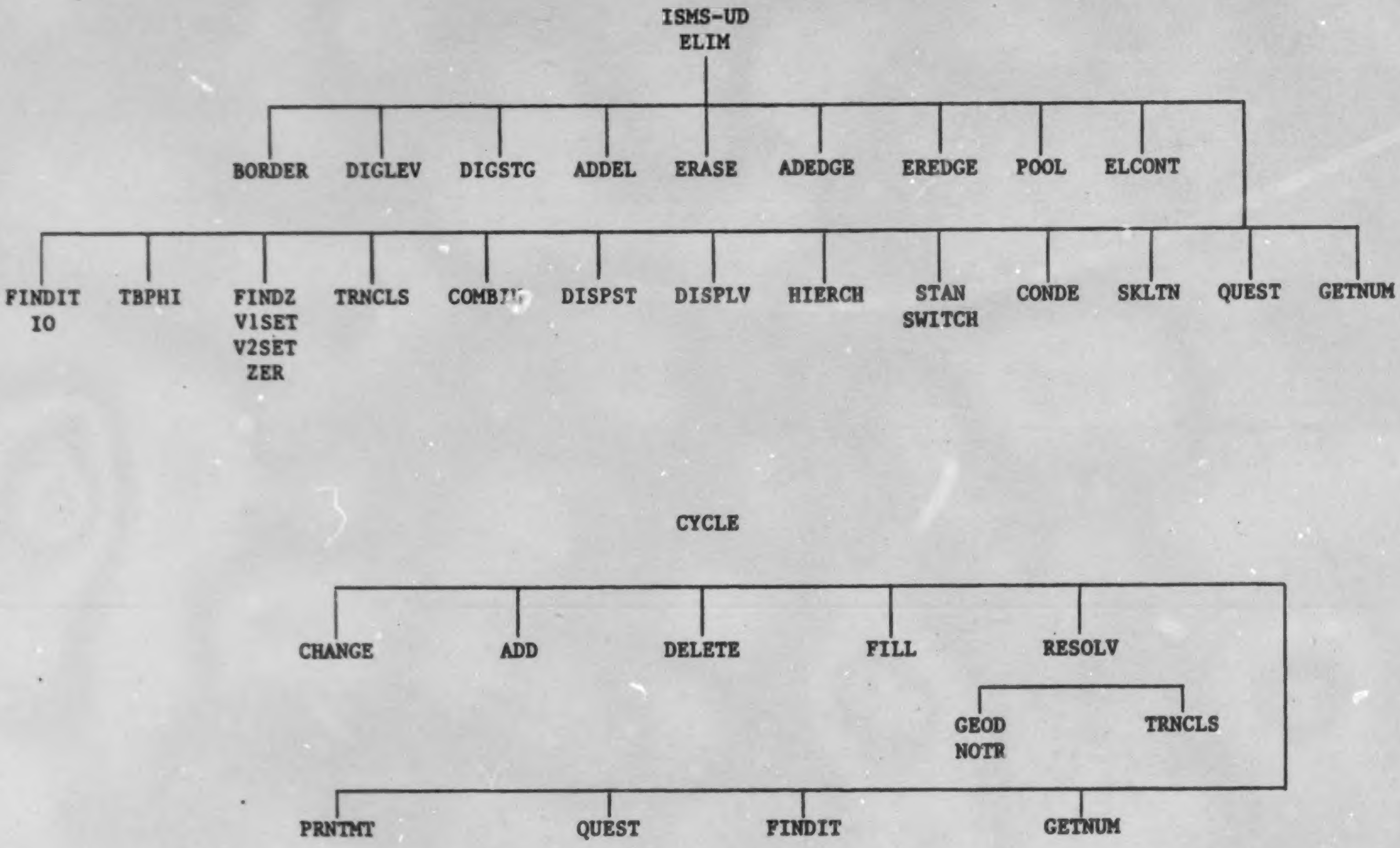

ISMS-UD
ELIM
BORDER
DIGLEV
DIGSTG
ADDEL
ERASE
ADEDGE
EREDGE
POOL
ELCONT
FINDIT
IO
TBPHI
FINDZ
V1SET
V2SET
ZER
TRNCLS
COMBIN
DISPST
DISPLV
HIERCH
STAN
SWITCH
CONDE
SKLTN
QUEST
GETNUM
CYCLE
CHANGE
ADD
DELETE
FILL
RESOLV
GEOD
NOTR
TRNCLS
PRNTMT
QUEST
FINDIT
GETNUM

Figure 8: Suggested ISMS-UD Overlay Scheme

Organizations that have Obtained ISMS-UD

Boeing Computer Services
20403-68th Avenue, South
Bldg. 18-43, Door S-4
Kent, Washington 98031

CACI, Inc.-Federal
1815 North Fort Myer Drive
Arlington, Virginia 22209

IBM Corporation
I/S Market Analysis Support
1000 Westchester Avenue
White Plains, New York 10604

Old Dominion University
Department of Electrical Engineering
Norfolk, Virginia 23508

Portland State University
630 SW Mill
Portland, Oregon 97207

Joe C. Roberts
1519 Meadowcrest
Garland, Texas 75042

Transportation Systems
GM Technical Center
12 Mile & Mound Roads
Warren, Michigan 48090

Tektronix, Inc.
Box 500
Beaverton, Oregon 97077

University of Ottawa
Computing Center, Operations
375 Nicholas Street, 3rd Floor
Ottawa, Ontario
K1N 6N5
CANADA

Bibliography

Fitz, Raymond, Joanne Troha, Karen Crim, Benjamin Gordon, and John Warfield. *Volume 3: Creating a Regional Environmental Learning System*. Dayton, OH: University of Dayton, 1979.

House, Robert, Andrew Sage, Raymond Fitz, David Yingling, Joanne Troha, and Karen Crim. *Volume 4: Conducting Collective Inquiry*. Nashville, TN: Vanderbilt University, 1979.

Knuth, Donald E. *The Art of Computer Programming*. Reading, MA: Addison-Wesley, 1973.

Warfield, John N. *Societal Systems: Planning, Policy, and Complexity*. New York, NY: Wiley-Interscience, 1976.

Attachment No. 1

DOCUMENTATION OF THE ISMS ALGORITHMS

Prepared by:

David R. Yingling, Jr.

This attachment provides a description of some of the algorithms used in ISMS-UD version 2. Although these algorithms have been comprehensively tested, the descriptions should be useful for debugging and future applications of ISM. The algorithms in this section are stated in the form used by Knuth (1973).

ISMS SUBROUTINE/FUNCTION NAME - BORDER

FUNCTION - Embeds elements into an existing reachability matrix via interactive questioning.

USAGE - CALL BORDER(N,RM,INDEX,TTYIN,TTYOUT, QTYPE,SUBREL,TXTWDS,SYS)

PARAMETERS

N - Input/Output integer scalar indicating the current number of elements in the argument reachability matrix.

RM - Input/Output logical two dimensional matrix of dimensions "SYS" X "SYS". This is the argument reachability matrix.

INDEX - Input/Output integer vector of length "SYS" containing the index set for "RM".

TTYIN - Input integer scalar used as unit number in FORTRAN READ statements directed to interactive terminal.

TTYOUT - Input integer scalar used as unit number in FORTRAN WRITE statements directed to interactive terminal.

QTYPE - Input logical variable indicating type of query to be presented.
QTYPE = .FALSE. causes full text queries to be used.
QTYPE = .TRUE. causes symbolic queries to be used.

SUBREL - Input logical variable indicating type of relationship to be modeled.
SUBREL = .FALSE. implies a non-subordinate relationship is being modeled.
SUBREL = .TRUE. implies a subordinate relationship is being modeled and questioning can therefore be optimized.

TXTWDS - Input integer scalar equal to (60/nchar) *10+10, where: nchar = number of display code characters able to be contained in one machine word.

SYS - Input integer scalar used in FORTRAN DIMENSION statements.

COMMON BLOCKS

FTEXT - Named integer common block of length 5*"TXTWDS".

COMMON PARAMETERS

R1 - Integer vector of dimension "TXTWDS".
R1(1-10) contain the number of machine words for each of the ten possible lines of the R1 clause
R1(11-end) contain the lines of text for the R1 clause. the text is packed with as many characters as possible in one machine word and each line starts on a word boundary.

L2 - Integer vector of length "TXTWDS". this vector is used as a scratch area.

R2 - Same attributes as "R1". the R2 clause is contained in this vector.

L2 - Same attributes as "L1".

R3 - Same attributes as "R1". the R3 clause is contained in this vector.

The common block "FTEXT" is used for full text queries. It is the calling routine's responsibility to fill vectors "R1", "R2", and "R3" if full text queries are desired.

REQUIRED ISMS ROUTINES - TBPHI,FINDZ,GETNUM,QUEST,IO,FINDIT

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - Transitive bordering on a reachability matrix. A description follows:

Algorithm TB (Transitive Bordering Algorithm). Given the number of elements currently in a logical matrix, m, the logical matrix, R, and the associated index set vectors, I, embed a new element, b, capitalizing on inference and transitivity.

TB1. [Initialize.] Set dimphi <-- n2. Set vector FLAG, flag[i] <-- .false., vector Z, z[i] <-- .false., matrix PHI, phi[i,j] <-- .false., i=1,2,...,dimphi, j=1,2,...,dimphi.

TB2. [Make sure physical bounds of arrays and vectors are not exceeded.] If (n + 1) <= FORTRAN dimension sizes then go to step TB3. Otherwise, type error message to interactive user and return to calling routine.

TB3. [Get value for b from interactive user.] Prompt user, call subroutine GETNUM, put value read in b, set i[n + 1] <-- b. If b=0 then return to calling program.

TB4. [Make sure that new value, b, is not duplicated.] Call subroutine FINDIT, if b has been already introduced into model matrix, issue error message and then go to step TB3.

TB5. [Make sure that textual information for b resides on text file.] If full text queries are not being used, skip immediately to step TB6. Read record from file. If no error condition is encountered, go to step TB6. Otherwise, issue error message and then go to step TB3.

TB6. [Form the transitive bordering inference opportunity matrix.] Form Phi (call subroutine TBPHI).

TB7. [Calculate the row/column of Phi with the maximum inference potential, zpoint.] Call subroutine FINDZ.

TB8. [Determine if a question filling the "y" portion of the reachability matrix is to be asked by examining the value of zpoint.] relate <-- zpoint. If zpoint > m, then go to step TB10.

TB9. [Ask for the "x" question and check for valid response.] Call subroutine QUEST(i[relate], i[m + 1]). Read response from user. zzz <-- .True.. If user typed "AB", go to step TB31, if user didn't type either a "Y" or "N", issue error message and then go to step TB9. If user typed "Y", zzz <--.false. and then go to step TB11. If user typed "N" then go to step TB12.

TB10. [Ask the "y" question and check for valid response.] Call subroutine QUEST(i[m + 1], i[zpoint - m]). Read response from user, zzz <-- .false.. If user typed "AB", go to step TB31, if user didn't type either a "Y" or "N", issue error message and then go to step TB10. If user typed "Y", zzz <-- .true. and then go to step TB11. If user typed "N" then go to step TB12.

TB11. [Process a yes answer to a query.] If zzz = .false., then go to step TB17, else go to step TB13.

TB12. [Process a no answer to a query.] If zzz = .false., then go to step TB17.

TB13. [Initialize.] Set j <-- 1.

TB14. [Search row of Phi for inference to be entered onto R.] If phi[zpoint, j] = .false., then go to step TB16.

TB15. [Fill up R with infered ones and zeros.] ictr2 <-- j. If ictr2 > n, then ictr2 <-- ictr2 - n. If j > n, then r[n + 1, ictr2] <-- .true.. Otherwise, r[ictr2, m + 1]<-- .false.. Set flag[j] <-- .true., z[j] <-- .true..

TB16. [Loop on j.] j <-- j + 1. If j <= dimphi, then go to step TB14, else go to step TB20.

TB17. [Initialize.] j <-- 1

TB18. [Search column of Phi for inference to be entered onto R.] if phi[j,zpoint] = .false., then go to step TB20.

TB19. [Fill up R with infered ones and zeros.] ictr2 <-- j, if ictr2 > n, then ictr2 <-- ictr2 - n. If j > n, then r[m + 1,ictr2] <-- .false., otherwise r[ictr2,n + 1] <-- .true.. Set flag[j] <-- .true., z(j) <-- .true.

TB20. [Loop on j.] j <-- j + 1. if j <= dimphi, then go to step TB18.

TB21. [Initialize.] j <-- 1

TB22. [Zero out rows and columns of Phi used in steps TB19 and TB15.] If z[j] = .false., then go to step TB24.

TB23. [Zero row and column j.] phi[j,k] <-- .false., phi[k,j] <-- .false., k=1,2,...,dimphi.

TB24. [Loop on j.] j <-- j + 1. If j <= dimphi, then go to step TB22.

TB25. [Initialize.] j <-- 1

TB26. [Check to see if Phi is all zeros via flag vector.] If flag[j] = .false., then go to step TB30.

TB27. [Loop on j.] j <-- j + 1. If j <= dimphi, then go to step TB26.

TB28. [Set main diagonal of R.] r[m + 1,m + 1] <-- .true., n <-- n + 1.

TB29. [Update permanent file with new R matrix.] Call subroutine IO, then go to step TB1.

TB30. [Zero out z.] z[j] <-- .false., j=1,2,...,dimphi. Go to step TB7.

TB31. [Write out message indicating abort of bordering session.] Write message and then go to step TB1.

TBPHI

ISMS SUBROUTINE/FUNTION NAME - TBPHI

FUNCION - Forms the Inference Opportunity Matrix used in the Transitive Bordering Algorithm.

USAGE - CALL TBPHI(N,A,DIMPHI,SUBREL,SYS2,SYS)

PARAMETERS

N - Input integer scalar indicating the current number of elements in matrix "A".

A - Input logical two dimensional matrix of dimension "SYS" X "SYS". This is the current reachability matrix model.

PHI - Output logical two dimensional matrix of dimension "SYS2" X "SYS2". This is the output inference opportunity matrix.

DIMPHI - Input integer scalar indicating the number of rows/columns in "PHI" which contain information. This variable should be set equal to 2*"N".

SUBREL - Input logical variable indicating type of relationship being modeled.
SUBREL = .FALSE. implies a non-subordinate relationship is being modeled.
SUBREL = .TRUE. implies a non-subordinate relationship is being modeled and the "PHI" matrix should be calculated differently.

SYS2 - Input integer scalar used in FORTRAN DIMENSION statements. "SYS2" is equal to 2*"SYS".

SYS - Input integer scalar used in FORTRAN DIMENSION statements.

COMMON BLOCKS - None

REQUIRED ISMS ROUTINES - None

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm FP (Form the Inference Opportunity Matrix, PHI, for Transitive bordering.) Given the number of elements in a reachability matrix, n, and the logical reachability matrix, A, form an inference opportunity matrix, PHI.

FP1. [Form the "n1" matrix.] Set PHI, phi[i,j] <-- a[i,j], phi[i + n,j + n] <-- a[i,j], phi[i + n,j] <-- .NOT.a[j,i], j=1,2,...,n, i=1,2,...,n.

FP2. [Process request for subordinate relationships.] Set PHI, phi[i + n,i] <-- .true., i=1,2,...,n if subrel is .true..

FP3. [Multiply.] Set phi[i,j + n] <-- .false. Then set phi[i,j + n] <-- phi[i,j + n] .OR.ed with the quantity phi[i + n,k] .AND.ed with phi[k + n,j + n], k=1,2,...,n, j=1,2,...,n, i=1,2,...,n.

FP4. [Multiply.] Set phi[i + n,j] <-- .false., then set phi[i + n,j] <-- phi[i + n,j] .OR.ed with the quantity phi[i,k] .AND.ed with phi[k,j + n], k=1,2,...,n, j=1,2,...,n, i=1,2,...,n.

FP5. [Set upper right half of matrix to zero.] Set phi[i,j] <-- .false., j=n+1,n+2,...,2*n, i=1,2,...,n.

FINDZ

ISMS SUBROUTINE/FUNCTION NAME		- FINDZ
FUNCTION		- Finds the row/column of an inference opportunity matrix that has maximum inference potential.
USAGE		- CALL FINDZ(DIMPHI,PHI,ZPOINT,FLAG,SYS2)
PARAMETERS	DIMPHI	- Input integer scalar indicating the number of rows and columns filled in the matrix "PHI".
	PHI	- Input logical two dimensional matrix of dimensions "SYS2" X "SYS2". This is the inference opportunity matrix.
	ZPOINT	- Output integer scalar which contains the row/column number in "PHI" with maximum inference potential. This value is returned to the calling routine.
	FLAG	- Input logical vector indicating rows/columns of "PHI" which have been zeroed out as a result of the transitive bordering process. A .true. indicates that a row and column has been zeroed out.
	SYS2	- Input integer scalar used in FORTRAN DIMENSION statements. "SYS2" is equal to 2*"SYS".
COMMON BLOCKS		- None
REQUIRED ISMS ROUTINES		- V1SET,V2SET
REQUIRED FORTRAN ROUTINES		- MINO,MAXO
ALGORITHM EMPLOYED		- The algorithm employed is described below:

Algorithm FZ (Find a value for Z that has the maximum inference potential.) Given the number of rows and columns in a logical matrix, dimphi, the logical matrix, PHI, and a logical vector indicating rows and columns of PHI that have been zeroed out, FLAG, find the row/column of PHI that has the maximum inference potential and set variable zpoint equal to this subscript.

FZ1. [Form the set V.] Set vector V1, v1[i] <-- number of ones in column i of PHI (apply algorithm V1). Set vector V2, v2[i] <-- number of ones in row i of PHI (apply algorithm V2). If flag[i] is .false., i=1,2,...,dimphi.

FZ2. [Form the set V′.] Set vector MIN, min[i] <-- the smaller of the two values, v1li] or v2[i] if flag[i] is .false., i=1,2,...,dimphi.

FZ3. [Find the maximum of the minimums of V′.] (Initialize.) Set bigger <-- 0, zpoint <-- 0.

FZ4. [Fill up vector Z.] Set big <-- the largest of the two values, bigger or min[i]. If big is greater than bigger, then zero out vector Z, z[k] <-- .false., k=1,2,...,i. If big is less than or equal to min[i], then set z[i] <-- .true., If big is greater than or equal to bigger, then set bigger <-- big, i=1,2,...,dimphi. Do the above step only if flag[i] = .false..

FZ5. [Form set V" and select a "Z".] Set bigger <-- 0. Set big <-- the greater of the two values, v1[i] + vz[i] or bigger. If bigger is greater than big, then set bigger <-- big and zpoint <-- i, i=1,2,...,dimphi. Do the above step if z[i] = .true..

V1SEI

ISMS SUBROUTINE/FUNCTION NAME - V1SET

FUNCTION - Counts the number of ones in a selected column, (J), of the inference opportunity matrix.

USAGE - X = V1SET(J,DIM,SYS2)

PARAMETERS

MAT - Input logical two dimensional matrix of dimension "SYS2" X "SYS2". This is the inference opportunity matrix.

J - Input integer scalar indicating the column of "MAT" that the counting operation should be applied to.

DIM - Input integer scalar indicating the last row/column of "MAT" that contains information.

SYS - Input integer scalar used in FORTRAN DIMENSION statements.

COMMON BLOCKS - None

REQUIRED ISMS ROUTINES - None

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm V1SET (Count the number of ones in the column of a matrix). Given the number of elements currently in a matrix, d, the matrix, M, and the number of a column in M, j, count the number of ones in column j of M and assign this value to s.

V1SET1. [Initialize.] Set s <-- 0, i <-- 1.

V1SET2. [Count.] If m[i,j] = 1, then s <-- s + 1.

V1SET3. [Loop on i.] i <-- i + 1. If i <= d, then go to step V1SET2, else algorithm is complete.

V2SET

ISMS SUBROUTINE/FUNCTION NAME - V2SET

FUNCTION - Counts the number of ones in a selected row (i) of the inference opportunity matrix.

USAGE - X = V2SET(MAT,I,DIM,SYS2)

PARAMETERS

MAT - Input logical two dimensional matrix of dimension "SYS2" X "SYS2". This is the inference opportunity matrix.

I - Input integer scalar indicating the row of "MAT" that the counting operation should be applied to.

DIM - Input integer scalar indicating the last row/column of "MAT" that contains information.

SYS2 - Input integer scalar used in FORTRAN DIMENSION statements. "SYS2" is equal to 2*"SYS".

COMMON BLOCKS - None

REQUIRED ISMS ROUTINES - None

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm V2SET (Count the number of ones in the row of a matrix). Given a logical matrix, M, the number of filled rows and columns in M, d, and the number of the row to count, i, count the total number of ones in row i of M and assign s to this value.

V2SET1. [Initialize.] s <-- 0, j <-- 1.

V2SET2. [Count.] If m[i,j] = 1, then set s <-- s + 1.

V2SET3. [Loop on j.] j <-- j + 1. If j <= d, then go to step V2SET2. Otherwise, algorithm is complete.

DIGLEV

ISMS SUBROUTINE/FUNCTION NAME - DIGLEV

FUNCTION - Displays a level formatted digraph of an argument reachability matrix.

USAGE - CALL DIGLEV(N,MATRIX,INDEX,TTYOUT,SYS,)

PARAMETERS

N - Input integer scalar denoting the current number of elements in the argument reachability matrix.

MATRIX - Input logical two dimensional matrix of dimension "SYS" X "SYS". This is the argument reachability matrix.

INDEX - Input integer vector of length "SYS" containing the index set for "MATRIX".

TTYOUT - Input integer scalar used as unit number in FORTRAN WRITE statements directed to interactive terminal.

SYS - Input integer scalar used in FORTRAN DIMENSION statements.

COMMON BLOCKS - None

REQUIRED ISMS ROUTINES - HIERCH,STAN,SWITH,CONDE,ELIM,SKLTN, DISPLV

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm DIGLEV (Display the levels formatted digraph of an argument reachability matrix). Given a reachability matrix, M, associated index set vector, I, and the current number of elements in M, n, display a levels formatted digraph on the interactive terminal.

DIGLEV1. [Rearrange M and I into hierarchial form. Leave result in H and O.] Apply Algorithm HEIRCH.

DIGLEV2. [Put H and O into standard form.] Apply Algorithm STAN.

DIGLEV3. [Calculate condensation matrix for H and O.] Apply Algorithm CONDE.

DIGLEV4. [Calculate skeleton matrix for H] Apply Algorithm SKLTN.

DIGLEV5. [Print out levels formatted digraph.] Apply Algorithm DISPLV.

DISPLV

ISMS SUBROUTINE/FUNCTION NAME - DISPLV

FUNCTION - Prints a levels formatted digraph from a lower triangular skeleton matrix.

USAGE - CALL DISPLV(N,SKLTN,INDEX,LEVELS,TTYOUT,SYS)

PARAMETERS

N - Input integer scalar indicating the number of elements in the input skeleton matrix.

SKLTN - Input logical two dimensional matrix of dimension "SYS" X "SYS". This is the input lower triangularized skeleton matrix.

INDEX - Input integer vector of length "SYS" containing the index set for "SKLTN".

LEVELS - Input integer vector of length "SYS" containing the number of elements on each level.
LEVELS(I) = Number of elements on each level #i.

TTYOUT - Input integer scalar used as unit number in FORTRAN WRITE statements directed to interactive terminal.

SYS - Input integer scalar used in FORTRAN DIMENSION statements.

COMMON BLOCKS - None

REQUIRED ISMS ROUTINES - None

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm DISPLV (Display a levels formatted digraph on the interactive terminal.) Given a logical lower triangularized skeleton matrix, S, its associated index set vector, D, the number of elements in S, n, and a vector filled with the number of elements on each level, L, display a levels formatted digraph.

```
DISPLV1.  [Initialize.] i <-- 1, level <-- 0, row <-- 1.

DISPLV2.  [Process next level.] level <-- level + 1, row <-- row + 1[level].  Write level number out on interactive terminal, variable level.

DISPLV3.  [Check to see if we are processing level one.] If level = 1, then go to step DISPLV11.

DISPLV4.  [Start processing element #i.] count <-- 0, iminus <-- i - 1, j <-- 1.

DISPLV5.  [Find all elements that element #i reaches to.] If s[i,J] = 0, then go to step DISPLV7.

DISLV6.   [Fill scratch print vector.] count <-- count + 1, list[count] <-- d[j].

DISPLV7.  [Loop on j.] j <-- j + 1.  If j <= iminus, then go to step DISPLV5.

DISPLV8.  [All done processing elements #i, print it and its connectives.] Write elements #i and its reachability set on interactive terminal, variables d[i], list[k], k=1,...,count.

DISPLV9.  [Are we finished with the algorithm?] i <-- i + 1.  If i < n, then algorithm is complete.

DISPLV10.  [Not done with algorithm, are we done with this level on digraph?] If i = row, then go to step DISPLV2.  Otherwise, go to step DISPLV3.

DISPLV11.  [Write out level #1 element on terminal.] Write out variable d[i], then go to step DISPLV9.
```

HIERCH

ISMS SUBROUTINE/FUNCTION NAME - HIERCH

FUNCTION - Rearranges an input reachability matrix into a levels partitioned hierarchial reachability matrix.

USAGE - CALL HIERCH(N,INREA,INDXIN,REAH,INDXH, NL,LEVEL,SYS)

PARAMETERS

N - Input integer scalar denoting number. elements in the input reachability matrix.

INREA - Input logical two dimensional matrix of dimensions "SYS" X "SYS". This is the input reachability matrix.

INDXIN - Input integer vector of length "SYS". This is the index set for "INREA".

REAH - Output logical two dimension matrix of dimensions "SYS" X "SYS". This is the output hierarchial levels partitioned matrix.

INDXH - Output integer vector of length "SYS". This is the index set for "REAH".

NL - Output integer scalar indicating the number of hierarchial levels in "REAH".

LEVEL - Output integer vector of length "SYS" containing the number of elements on each level.
LEVELS(I) = Number of elements on level #i.

SYS - Input integer scalar used in FORTRAN DIMENSION statements.

COMMON BLOCKS - None

REQUIRED ISMS ROUTINES - None

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm HIERCH (Rearrange an input reachability matrix into a levels partitioned hierarchial reachability matrix.) Given a reachability matrix, R, its associated index set vector, D, the number of elements in R, n, rearrange R according to a levels partition algorithm. Leave this result in an output reachability matrix, H, and associated index set, E.

HIERCH1. [Initialize.] Copy R into H. Set local vector F to zero. h[i,j] <---r[i,j], f[i] <-- 0, j=1,2,....,n, i=1,2,....,n.

HIERCH2. [Initialize levels partition algorithm.] nl <-- 0, neap <-- 0.

HIERCH3. [Begin levels partition algorithm.] nl <-- nl + 1, nel <-- 0.

HIERCH4. [Find an element to process.] i <-- 1.

HIERCH5. [Skip past element row/column already processed.] If f[i] = 1, then go to step HIERCH10.

HIERCH6. [Initialize subset test.] i <-- 1.

HIERCH7. [Test to see if reachable set is not a subset of antecedent set for element #i.] If r[i,j] = 1 .AND. r[j,i] = 0, then go to step HIERCH10.

HIERCH8. [Loop on j.] j <-- j + 1. If j <= n, then go to step HIERCH7.

HIERCH9. [Fill up E.] nel <-- nel + 1, e[neap + nel] <-- d[i], local vector T, t[nel] <-- i.

HIERCH10. [Loop on i.] i <-- i + 1. If i <= n, then go to step HIERCH5.

HIERCH11. [Found all elements on this level.] neap <-- neap + nel, vector levels, level[nl] <-- nel.

```
HIERCH12.  [Initialize i] i <-- 1.
HIERCH13.  [Identify a row/column to be zeroed.] x <-- t[i], f[x] <-- 1.
HIERCH14.  [Initialize zero out routine.] j <-- 1.
HIERCH15.  [Zero out R.] h[x,j] <-- 0, h[j,x] <-- 0.
HIERCH16.  [Loop on j.] j <-- j + 1.  If j <= n, then go to step HIERCH15.
HIERCH17.  [Loop on i.] i <-- i + 1.  If i <= nel, then go to step HIERCH13.
HIERCH18.  [Have all elements been processed?] If near < n, then go to step HIERCH12.
HIERCH19.  [Initialize i.] i <-- 1.
HIERCH20.  [Initialize j.] j <-- 1.
HIERCH21.  [Locate proper j subscript.] If e[i] = d[j], then go to step HIERCH23.
HIERCH22.  [Loop on j.] j <-- j + 1.  If j <= n, then go to step HIERCH21.
HIERCH23.  [Initialize k.] k <-- 1.
HIERCH24.  [Rearrange rows of R and store result in H.] h[i,k] <-- r[j,k].
HIERCH25.  [Loop on k.] k <-- k + 1.  If k <= n, then go to step HIERCH24.
HIERCH26.  [Loop on i.] i <-- i + 1.  If i <= n, then go to step HIERCH20.
HIERCH27.  [Copy H into R.] r[i,j] <-- h[i,j], j=1,2,...,n, i=1,2,...,n.
HIERCH28.  [Initialize i.] i <-- 1.
HIERCH29.  [Initialize j.] j <-- 1.
HIERCH30.  [Locate proper j subscript.] If e[i] = d[j], then go to step HIERCH32.
```

HIERCH31. [Loop on j.] j <-- j + 1. If j <= n, then go to step HIERCH30.

HIERCH32. [Initialize k.] k <-- 1.

HIERCH33. [Rearrange columns of R and store in in H.] h[k,i] <-- r[k,j]

HIERCH34. [Loop on k.] k <-- k + 1. If k <= n, then go to step HIERCH33.

HIERCH35. [Loop on i.] i <-- i + 1. If i => n, then go to step HIERCH29.

HIERCH36. [Copy E into D, copy H into R.] d[i] <-- e[i], r[i,j] <-- h[i,j], j=1,2,...,n, i=1,2,...,n.

STAN

ISMS SUBROUTINE/FUNCTION NAME - STAN

FUNCTION - Converts an input hierarchial reachability matrix into its standard form.

USAGE - CALL STAN(N,MATRIX,INDEX,NLEVEL,LEVELS,SYS)

PARAMETERS

N - Input integer scalar denoting the number of elements in "MATRIX".

MATRIX - Input hierarchial reachability matrix, output standard form matrix. This is a logical two dimensional matrix with dimensions "SYS" X "SYS".

INDEX - Input/Output vector of length "SYS" containing the index set for matrix.

NLEVEL - Input integer scalar denoting the number of hierarchial levels in "MATRIX".

LEVELS - Input integer vector of length "SYS" containing the number of elements on each level.
LEVELS(I) = Number of elements on level #I.

SYS - Input integer scalar used in FORTRAN DIMENSION statements.

COMMON BLOCKS - None

REQUIRED ISMS ROUTINES - SWITCH

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm STAN (Compute standard form matrix). Given a hierarchial reachability matrix, H, its associated index set vector, V, the number of elements in H, n, the number of hierarchial levels on H, nlevel, and a vector containing the number of elements on each level, L, compute the standard form of H.

```
STAN1.  [Initialize.] end <-- 0, i <-- 1.
STAN2.  [Continue to check to make sure all non-cycle elements are on top.] start <-- end + 1, end <-- end + l[i]
STAN3.  [Check to see if number of elements is two or less.] If l[i] .LE. 2, then go to step STAN13.
STAN4.  [Find and move all non-cycle elements together on each level.] ii <-- start
STAN5.  [Initialize row check.] row <-- start, last <-- 0.
STAN6.  [Initialize column check.] col <-- start, nones <-- 0.
STAN7.  [Count number of ones.] If h[row,col] = 1, then nones <-- nones + 1.
STAN8.  [Loop on col.] col <-- col + 1.  If col .LE.  end, then go to step STAN7.
STAN9.  [Check to see if a switch should be performed.] If nones .LT. last, then apply algorithm SWITCH.
STAN10.  [Reset variable last.] If nones .GE.  last, then set last <-- nones.
STAN11.  [Loop on row.] row <-- row + 1.  If row .LE.  end, then go to step STAN6.
STAN12.  [Loop on ii.] ii <-- ii + 1.  If ii .LE.  end, then go to step STAN5.
STAN13.  [Loop on i.] i <-- i + 1.  If i .LE..  nlevel, then go to step STAN2.
STAN14.  [Calculate variable nminus.] nminus <--n - 1.
STAN15.  [Set switch flag to 0.] swit <-- 0.
STAN16.  [Initialize i.] i <-- 1.
STAN17.  [Initialize j.] iplus <-- i + 1, j <-- iplus.
```

STAN18. [Check for ones above main diagonal.] If h[i,j] = 0, then go to step STAN21.

STAN19. [Check to see if a switch is needed.] jminus <-- j - 1. If i = jminus, then go to step STAN22.

STAN20. [Apply algorithm SWITCH to switch row and col j with row and col j-1.] Apply algorithm SWITCH. Set swit <-- 1, j <-- j - 1, then go to step STAN19.

STAN21. [Loop on j.] j <-- j + 1. If j .LE. n, then go to step STAN18.

STAN22. [Loop on i.] i <-- i + 1. If i .LE. nminus, then go to step STAN17.

STAN23. [Check to see if any switching was required.] If swit = 1, then go to step STAN15.

ISMS SUBROUTINE/FUNCTION NAME - SKLTN

FUNCTION - Calculates the nonredundant adjacency matrix (skeleton matrix) given a condensation matrix.

USAGE - CALL SKLTN(N,MATRIX,SYS)

PARAMETERS N - Input integer scalar denoting the number of elements in "MATRIX".

MATRIX - Input/Output logical two dimensional matrix of dimensions "SYS" X "SYS". This is the input condensation matrix/ output skeleton matrix.

SYS - Input integer scalar used in FORTRAN DIMENSION statements.

COMMON BLOCKS - None

REQUIRED ISMS ROUTINES - None

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is similar to the one described by R.K. Shyamasundar in IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-8, No. 2, February, 1978. It is described below.

Algorithm SKLTN (Calculate the nonredundant adjacency matrix). Given a condensation matrix, C, and the number of elements contained in C, n, calculate the skeleton matrix.

SKLTN1. [Initialize.] nminus <-- n 1, i <-- 2.

SKLTN2. [Initialize.] iminus <-- i - 1, j <-- 1.

SKLTN3. [Check reachability of node j to node i.] If c[i,j] = 0, then go to step SKLTN7.

SKLTN4. [Initialize.] iplus <-- i + 1, k <-- iplus.

SKLTN5. [Add all nodes to row i that can be reached from node i.] If c[k,j] = 1. .AND.c[k,i] = 1, then c[k,j] = 0.

SKLTN6. [Loop on k.] k <-- k + 1. If k .LE. n, then go to step SKLTN5.

SKLTN7. [Loop on j.] j <-- j + 1. If j .LE. iminus, then go to step SKLTN3.

SKLTN8. [Loop on i.] i <-- i + 1. If i .LE. nminus, then go to step SKLTN2.

ISMS SUBROUTINE/FUNCTION NAME - SWITCH

FUNCTION - Exchanges two rows and columns on a binary martix.

USAGE - CALL SWITCH(N,MATRIX,INDEX,ROW,SYS)

PARAMETERS

N - Input integer scalar denoting the number of elements in "MATRIX".

MATRIX - Input/Output logical two dimensional matrix of dimensions "SYS" X "SYS".

INDEX - Input/Output vector of length "SYS" containing the index set of "MATRIX"

SYS - Input integer scalar used in FORTRAN DIMENSION statements.

COMMON BLOCKS - None

REQUIRED ISMS ROUTINES - None

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm SWITCH (Exchange two rows and columns of a binary matrix). Given a binary matrix, M, its associated index set, R, the number of elements in M, n, and the number of a row and column to be switched; s, switch row and column s with row and column s-1.

```
SWITCH1.  [Initialize.] other <-- s - 1, i <-- 1.

SWITCH2.  [Switch rows.] temp <-- m[s,i], m[s,i] <-- m[other,i],
        m[other,i] <-- temp.

SWITCH3.  [Loop on i.] i <-- i + 1.  If i .LE.  n, then go to step
        SWITCH2.
```

```
SWITCH4.  [Initialize.] i <-- 1.
SWITCH5.  [Switch the columns.] temp <--m[i,s], m[i,s] <-- m[i,other], m[i,other] <-- temp.
SWITCH6.  [Loop on i.] i <-- i + 1.  If i .LE.  n, then go to step SWITCH5.
SWITCH7.  [Switch index set.] temp <-- r[s], r[s] <-- r[other], r[other] <-- temp.
```

ISMS FUNCTION/SUBROUTINE NAME - ZER

FUNCTION - Zeros out a logical vector.

USAGE - CALL ZER(VECTOR,I,SYS)

PARAMETERS VECTOR - Input/Output logical vector of length "SYS".

I - Input integer scalar denoting starting index of values to be retained in "VECTOR".

SYS - Input integer scalar used in FORTRAN DIMENSION statements.

COMMON BLOCKS - None

REQUIRED ISMS ROUTINES - None

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm ZER (Zero out selected positions on a logical vector). Given a logical vector, V, and the value of the first index of V to retain, i, zero out the vector.

ZER1. [Initialize.] ii <-- i - 1, j <-- 1.

ZER2. [Zero out.] v[j] <-- 0.

ZER3. [Loop on j.] j <-- j + 1. If j .LE. ii, then go to step ZER2.

ISMS SUBROUTINE/FUNCTION NAME - IO

FUNCTION - Reads and writes ISMS information on disk file.

USAGE - CALL IO(N,MAT,INDEX,UNITNO,READ,SYS)

PARAMETERS

- N - Input/Output integer scalar contains the number of elements in "MAT".
- MAT - Input/Output logical two dimensional matrix of dimensions "SYS" X "SYS". This is the ISM binary matrix.
- INDEX - Input/Output integer vector of length "SYS". This is the index set vector for "MAT".
- UNITNO - Input integer scalar used as unit number in FORTRAN READ and WRITE statements for disk file.
- READ - Input logical scalar used for determining if a read or write operation is desired.
  READ = .TRUE. means a read operation is requested.
  READ = .FALSE. means a write operation is requested.
- SYS - Input integer scalar used in FORTRAN DIMENSION statements.

COMMON BLOCKS - None

REQUIRED ISMS ROUTINES - None

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm IO (Read and write ISM information on disk file). Given a logical switch variable, r, read or write the ISM binary matrix, index set, and number of elements.

IO1. [Determine if a read or write operation is desired.] If r = 1, then go to step IO3.

IO2. [Write operation.] Rewind file. Write n, MAT, and INDEX. Algorithm is complete.

IO3. [Read operation.] Rewind file. Read n, MAT, and INDEX. Algorithm is complete.

DISPSI

ISMS SUBROUTINE/FUNCTION NAME		- DISPST
FUNCTION		- Prints a stages formatted digraph given an upper triangularized skeleton matrix.
USAGE		- CALL DISPST(N,MATRIX,INDEX,STAGES,NSTAGE,TTYOUT,SYS)
PARAMETERS	N	- Input integer scalar equal to the number of elements in "MATRIX".
	MATRIX	- Input logical two dimensional matrix of dimensions "SYS" X "SYS". This is the upper triangularized skeleton matrix.
	INDEX	- Input integer vector of length "SYS" containing the index set of "MATRIX".
	STAGES	- Input integer vector of length "SYS" containing the number of elements on each stage. STAGES(I) = Number of elements on stage #I.
	NSTAGE	- Input integer scalar equal to the total number of stages on "MATRIX".
	TTYOUT	- Input integer scalar used as unit number in FORTRAN WRITE statements directed to interactive terminal.
	SYS	- Input integer scalar used in FORTRAN DIMENSION statements.
COMMON BLOCKS		- None
REQUIRED ISMS ROUTINES		- None
REQUIRED FORTRAN ROUTINES		- None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm DISPST (Print a stages formatted digraph). Given the number of elements currently in an upper triangularized skeleton matrix, n, the skeleton matrix, S, the index set for S, T, a vector containing the number of elements on each stage, R, and the total number of stages, v, print the stages formatted digraph.

DISPST1. [Initialize.] row <-- 1, stage <-- 0, i <-- 1.

DISPST2. [Initialize this stage.] stage <-- stage + 1, row <-- row + r[stage]. Write out stage number on interactive terminal, variable stage.

DISPST3. [Check to see if last stage is to be processed.] If stage = v, then go to step DISPST13.

DISPST4. [Initialize element search.] count <-- 0, istart <-- i + 1, j <-- istart.

DISPST5. [Find all elements that element #i reaches.] If s[i,j] = 0, then go to step DISPST7.

DISPST6. [Found one, keep a record of it.] count <-- count + 1, fill local vector LIST, l[count] <-- t[j].

DISPST7. [Loop on j.] j <-- j + 1. If j .LE. n, then go to step DISPST5.

DISPST8. [Special printing routine for elements with no reachability set.] If count = 0, then go to step DISPST13.

DISPST9. [All done processing element #i, print on terminal.] Write out on interactive terminal, t[i], l[k], k=1,2,...,count.

DISPST10. [Increment i.] i <-- i + 1.

DISPST11. [Finished with algorithm?] If i .GT. n, then algorithm is complete.

DISPST12. [Finished with this stage?] If i = row, then go to step DISPST2, else go to step DISPST3.

DISPST13. [Process last stage elements.] Write out on interactive terminal t[i], then go to step DISPST10.

FINDIT

ISMS SUBROUTINE/FUNCTION NAME		- FINDIT
FUNCTION		- Determines if two elememts are contained in the index set of a matrix.
USAGE		- CALL FINDIT(N,N1,N2,S1,S2,INDEX,FOUND1, FOUND2,SYS)
PARAMETERS	N	- Input integer scalar denoting number of elements contained in "INDEX".
	N1	- Input integer scalar equal to the first element number to be found.
	N2	- Input integer scalar equal to the second element number to be found.
	S1	- Output integer scalar which is set equal to the subscript of "INDEX" that "N1" is found.
	S2	- Output integer scalar which is set equal to the subscript of "INDEX" that "N2" is found.
	INDEX	- Input integer vector of length "SYS". This is an index set vector for a matrix.
	FOUND1	- Output logical scalar set to .true. if "N1" was found, set to .false. otherwise.
	FOUND2	- Output logical scalar set to .true. if "N2" was found, set to .false. otherwise.
	SYS	- Input integer scalar used in FORTRAN DIMENSION statements.
COMMON BLOCKS		- None

REQUIRED ISMS ROUTINES - None

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm FINDIT (Find the subscript values for two elements in any index set vector). Given two element numbers to be found, n1 and n2, the number of elements in a given index set, n, the index set vector, I, find the vector index for each, s1 and s2, and set a logical variable for each, f1 and f2, if found.

FINDIT1. [Initialize.] f1 <-- 0, f2 <-- 0, s1 <-- 0, s2 <-- 0, j <-- 1.

FINDIT2. [Search.] If n1 = i[j], then set s1 <-- j. If n2 <-- i[j], then set s2 <-- j.

FINDIT3. [Loop on j.] j <-- j + 1. If j .LE. n, then go to step FINDIT2.

FINDIT4. [Set logical variables.] If s1.gt. 0, then set f1 <-- 1. If s2 .GT. 0, then set f2 <-- 1.

DIGSIG

ISMS SUBROUTINE/FUNCTION NAME - DIGSTG

FUNCTION - Displays a staged formatted digraph from a given reachability matrix.

USAGE - CALL DIGSTG(N,MATRIX,INDEX,TTYOUT,SYS)

PARAMETERS

N - Input integer scalar denoting the number of elements in "MATRIX".

MATRIX - Input logical two dimensional matrix of dimensions "SYS" X "SYS". This is the input reachability matrix.

INDEX - Input integer vector of length "SYS". This is the index set for "MATRIX".

TTYOUT - Input integer scalar used as unit number in FORTRAN WRITE statements directed to interactive terminal.

SYS - Input integer scalar used in FORTRAN DIMENSION statements.

COMMON BLOCKS - None

REQUIRED ISMS ROUTINES - HIERCH,STAN,SWITCH,CONDE,ELIM,SKLTN, DISPST

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm DIGSTG (Display the digraph of a reachability matrix in a stages format). Given a reachability matrix, R, its associated index set, X, and the number of elements in R, n, display the digraph in a stages format.

DIGSTG1. [Check for error condition.] If n .GT. 0, then go to step DIGSTG3.

DIGSTG2. [Write out error message.] Write error to interactive terminal and then algorithm is complete.

DIGSTG3. [Initialize.] i <-- 1.

DIGSTG4. [Initialize.] j <-- 1.

DIGSTG5. [Transpose input reachability matrix.] Set local logical matrix, I. t[i,j] <-- r[j,i].

DIGSTG6. [Loop on j.] j <-- j + 1. If j .LE. n, then go to step DIGSTG5.

DIGSTG7. [Loop on i.] i <-- i + 1. Ifi .LE. n then go to step DIGSTG4.

DIGSTG8. [Copy T into R.] r[i,j] <-- t[i,j], j=1,2,...,n, i=1,2,...,n.

DIGSTG9. [Levels partition T.] Apply Algorithm HIERCH.

DIGSTG10. [Put T into standard form.] Apply Algorithm STAN.

DIGSTG11. [Compute condensation matrix of T.] Apply Algorithm CONDE.

DIGSTG12. [Calculate skeleton matrix of T.] Apply Algorithm SKLTN.

DIGSTG13. [Initialize.] i <-- 1.

DIGSTG14. [Initialize.] j <-- 1.

DIGSTG15. [Transpose T, leave result in R.] r[i,j] <-- t[j,i].

DIGSTG16. [Loop on j.] j <-- j + 1. If j .LE. n, then go to step DIGSTG15.

DIGSTG17. [Loop on i.] i <-- i + 1. If i .LE. n, then go to step DIGSTG14.

DIGSTG18. [Print out stages digraph.] Apply Algorithm DISPST.

ELCONT

ISMS SUBROUTINE/FUNCTION NAME		- ELCONT
FUNCTION		- Performs the elementary contraction process.
USAGE		- CALL ELCONT(N,REA,INDEX,TTYIN,TTYOUT,SYS)
PARAMETERS	N	- Input/Output integer scalar equal to the number of elements in "REA".
	REA	- Input/Output logical two dimensional matrix of dimensions "SYS" X "SYS".
	INDEX	- Input/Output integer vector of length "SYS" containing the index set of "REA".
	TTYIN	- Input integer scalar used as unit number in FORTRAN READ statements directed to interactive terminal.
	TTYOUT	- Input integer scalar used as unit number in FORTRAN WRITE statements directed to interactive terminal.
	SYS	- Input integer scalar used in FORTRAN DIMENSION statements.
COMMON BLOCKS		- None
REQUIRED ISMS ROUTINES		- GETNUM,FINDIT,HIERCH,STAN,SWITCH,CONDE,ELIM,SKLTN,COMBIN,TRNCLS,IO
REQUIRED FORTRAN ROUTINES		- None
ALGORITHM EMPLOYED		- The algorithm employed is described below:

Algorithm ELCONT (Contract two adjacent elements on different levels or stages). Given a reachability matrix, R, its associated index set vectors, T, and the number of elements in R, n, ask the user for two elements and perform the elementary contraction process.

ELCONT1. [Accept two number from interactive terminal.] Apply Algorithm GETNUM.

ELCONT2. [Check to see if numbers are zero.] If u = 0 or v = 0, then algorithm is complete.

ELCONT3. [Check to see if numbers are in index set.] Apply Algorithm FINDIT. If u and/or v are/is not found, then write appropriate error message on interactive terminal and then go to step ELCONT1.

ELCONT4. [Set nn.] nn <-- n.

ELCONT5. [Calculate non-redundant adjacency matrix for "REA".] Apply Algorithm HIERCH, Algorithm STAN, Algorithm CONDE, Algorithm SKLTN. Fill up local vector Z with index set of non-redundant adjacency matrix and local matrix A with matrix.

ELCONT6. [Check to see if u and v are adjacent.] Apply Algorithm FINDIT in order to obtain the row and column subscripts for u and v, iu, iv. If u and v are found on A and a[iu,iv] = 1, then go to step ELCONT8.

ELCONT7. [Element u is not adjacent to element v.] Write appropriate error message on interactive terminal and then go to step ELCONT1.

ELCONT8. [Accept new index value from interactive terminal for contracted elements.] Apply Algorithm GETNUM. If newnam = 0, then the algorithm is complete.

ELCONT9. [Check to see if newnam is already in index set.] Apply Algorithm FINDIT. If newnam is already used, then write appropriate error message on interactive terminal and then go to step ELCONT8.

ELCONT10. [Combine elements u and v and use newnam as index value.] Apply Algorithm COMBIN.

ELCONT11. [Calculate reachability of new element.] Apply Algorithm TRNCLS.

ELCONT12. [Update permanent file with new matrix.] Apply Algorithm IO and then go to step ELCONT1.

CONDE

ISMS SUBROUTINE/FUNCTION NAME - CONDE

FUNCTION - Computes the condensation matrix of a given standard form matrix.

USAGE - CALL CONDE(N,MATRIX,INDEX,LEVELS,TTYOUT,TYPE,SYS)

PARAMETERS

N - Input/Output integer scalar denoting the number of elements in the input standard form matrix/number of elements in the output condensation matrix.

MATRIX - Input/Output logical two dimensional matrix of dimensions "SYS" X "SYS". This is the input standard form matrix/output condensation matrix.

INDEX - Input/Output integer vector of length "SYS". This is the index set of the input standard form matrix/output condensation matrix.

LEVELS - Input/Output integer vector of length "SYS". This vector contains the number of elements on each level of the input standard form matrix/output condensation matrix.

TTYOUT - Input integer scalar used as unit number in FORTRAN WRITE statements directed to interactive terminal.

TYPE - Input logical scalar used to determine if the printing of cycles at the interactive terminal is to be performed.
TYPE = .TRUE. means print cycles.
TYPE = .FALSE. means not to print cycles.

SYS - Input integer scalar used in FORTRAN DIMENSION statements.

COMMON BLOCKS - None

REQUIRED ISMS ROUTINES - ELIM

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm CONDE (Compute the condensation matrix of a given standard form matrix.) Given a standard form matrix, S, its associated index set vector, R, the number of elements in S, n, a vector with the number of elements on each level on S, L, and a variable that tells whether or not to print the cycles, b, reduce all cycle sets to a single proxy element by eliminating elements to finally yield the condensation matrix.

CONDE1. [Initialize i.] i <-- 1.

CONDE2. [Initialize.] count <-- 1, j <-- i + 1, local scratch vector T, t[1] <-- r[ ].

CONDE3. [Check for a one above main diagonal (cycle).] If s[i,j] = 0, then go to step CONDE11.

CONDE4. [Put found cycle element in print out list and eliminate from matrix.] count <-- count + 1, t[count] <-- r[j], apply algorithm ELIM.

CONDE5. [Initialize.] positn <-- 0, ii <-- 1.

CONDE6. [Find proper element to change in L.] positn <-- positn + l[i]. If positn .GE. j, then go to step CONDE8.

CONDE7. [Loop on ii.] ii <-- ii + 1. If ii .LE. n, then go to step CONDE6.

CONDE8. [Reduce number of elements on level where an element was eliminated.] l[ii] <-- l[ii] - 1.

CONDE9. [Any more elements in this cycle set?] If s[i,j] = 1 .AND. j .LE. n, then go to step CONDE4.

CONDE10. [Write cycle out to interactive terminal.] If b = 1, then write out t[iii], iii=1,2,...,count.

CONDE11. [Loop on i.] i <-- i + 1. If i .LT. n, then go to step CONDE2.

ISMS SUBROUTINE/FUNCTION NAME		- GETNUM
FUNCTION		- Reads n unsigned integers from an interactive terminal in a free format.
USAGE		- CALL GETNUM(ARRAY,N,TTYIN,TTYOUT)
PARAMETERS	ARRAY	- Output integer vector of length "N". This vector contains the number as read from the terminal ARRAY(1) = First number, etc.
	N	- Input integer equal to the number of integers to be read.
	TTYIN	- Input integer scalar used as unit number in FORTRAN READ statements directed to interactive terminal.
	TTYOUT	- Input integer scalar used as unit number in FORTRAN WRITE statements directed to interactive terminal.
COMMON BLOCKS		- None
REQUIRED ISMS ROUTINE		- None
REQUIRED FORTRAN ROUTINES		- None
ALGORITHM EMPLOYED		- The algorithm employed is described below:

Algorithm GETNUM (Read "n" unsigned integers in a free format). Given the vector to store the integers, A, the number of integers to read, n, read "n" unsigned integers from the interactive terminal.

```
GETNUM1.  [Read in string from terminal.] Read into local vector B.

GETNUM2.  [Initialize.] l <-- n, a[l] <-- 0, power <-- 0, i <-- 1.
```

GETNUM3. [Search backwards through string looking for deliminters.] k <-- 81 - 1. If b[k] = blank or a comma, then go to step GETNUM9.

GETNUM4. [Initialize.] j <-- 1, ii <-- j - 1.

GETNUM5. [Check against holerith constant vector (nums).] If b[k] .NE. nums[j], then go to step GETNUM7.

GETNUM6. [Construct integer.] a[l] <-- a[l] + (ii * (10**power)), power <-- power + 1, then go to step GETNUM12.

GETNUM7. [Loop on j.] j <-- j + 1. If j .LE. 10, then go to step GETNUM5.

GETNUM8. [Character found was not numeric.] Write error message and then go to step GETNUM1.

GETNUM9. [Check for end of number.] If a[l] = 0, then go to step GETNUM12.

GETNUM10. [Decrement l and check for completion.] l <-- l - 1. If l = 0, then go to step GETNUM13.

GETNUM11. [Initialize.] a[l] <-- 0, power <-- 0.

GETNUM12. [Loop on i.] i <-- i + 1. If i .LE. 80, then go to step GETNUM3.

GETNUM13. [Initialize.] i <-- 1.

GETNUM14. [Check magnitude of each number.] If a[i] .GT. 99999, then go to step GETNUM16.

GETNUM15. [Loop on.] i <-- i + 1. If i .LE. n, then go to step GETNUM14. Otherwise, algorithm is complete.

GETNUM16. [Numbers too large.] Write out error message and then go to step GETNUM1.

ISMS SUBROUTINE/FUNCTION NAME - COMBIN

FUNCTION - Combines two rows and columns in a reachability matrix.

USAGE - CALL COMBIN(N,REA,INDEX,IU,IV,NEWNAM, SYS)

PARAMETERS

N - Input/Output integer scalar equal to the number of elements in "REA".

REA - Input/Output logical two dimensional matrix of dimensions "SYS" X "SYS". This is the argument reachability matrix.

INDEX - Input/Output integer vector of length "SYS" containing the index set of "REA".

IU - Input integer scalar equal to the row/column subscript of the first elements to be combined.

IV - Input integer scalar equal to the row/column subscript of the second element to be combined.

NEWNAM - Input integer scalar equal to the integer to be used in the index set for the combined elements.

SYS - Input integer scalar used in FORTRAN DIMENSION statements.

COMMON BLOCKS - None

REQUIRED ISMS ROUTINES - ELIM

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm COMBIN (Replace row and column iv with the boolean sum of row and column iu and iv). Given the number of elements in a reachability matrix, n, the reachability matrix, R, its associated index set vector, T, the two row/column indices to combine, iu and iv, and the value to be replaced in the index set for the combined element, newnam, combine iu and iv via boolean sum method and replaced index name with newnam.

COMBIN1. [Initialize.] i <-- 1.

COMBIN2. [Replace row iv with boolean sum of rows iu and iv.] r[iv,i] <-- r[iu,i] .OR. r[iv,i].

COMBIN3. [Replace column iv with boolean sum of columns iu and iv.] r[i,iv] <-- r[i,iu] .OR. r[i,iv].

COMBIN4. [Loop on i.] i <-- i + 1. If i .LE. n, then go to step COMBIN2.

COMBIN5. [Replace iv's index with newnam.] t[iv] <-- newnam.

COMBIN6. [Erase row, column, and index for iu.] Apply Algorithm ELIM.

ISMS SUBROUTINE/FUNCTION NAME - QUEST

FUNCTION - Displays the queries used during an embedding session with ISM.

USAGE - CALL QUEST(EL1,EL2,TTYOUT,QTYPE,TXTWDS)

PARAMETERS

EL1 - Input integer scalar which is equal to the index value of the first element to be displayed.

EL2 - Input integer scalar which is equal to the index value of the second element to be displayed.

TTYOUT - Input integer scalar used as unit number in FORTRAN WRITE statements directed to interactive terminal.

QTYPE - Input integer scalar used to determine if full text or symbolic queries
QTYPE = .TRUE. means that symbolic queries will be used.
QTYPE = .FALSE. means that full text queries will be used.

TXTWDS - Input integer scalar which is equal to the number of machine words required to hold 600 display code characters plus 10.

COMMON BLOCKS FTEXT - Named integer common block

See the description under subroutine BORDER for "FTEXT" parameters

REQUIRED ISMS ROUTINES - None

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm QUEST (Display the ISM queries in either full text or symbolic formats). Given the index values of each element to be displayed, e1 and e2, a switch variable which determines what type of queries to use, q, and a common block filled with the introductory, relational, and qualifying phrases, display the query.

QUEST1. [Determine query type to be presented.] If q = 1, then go to step QUEST12.

QUEST2. [Calculate direct access file offset.] i1 <-- e1 + 4, i2 <-- e2 + 4.

QUEST3. [Read elements' text file and put in common block.] Read record i1 and place in "L1". Read record i2 and place in "L2".

QUEST4. [Initialize.] offset <-- 0, i <-- 1.

QUEST5. [Initialize.] i1 <-- 0, i2 <-- 0, j <-- 1.

QUEST6. [Check length indicator.] (NOTE: a local vector, B, is equivalenced to the common block.) If b[j + offset] = 0, then go to step QUEST9.

QUEST7. [Compute length and location of print line.] length <-- b[j + offset], i1 <-- i2 + 1, i2 <-- i1 + length - 1.

QUEST8. [Print line of interactive terminal.] Write out b[c + offset + 10], c=i1,...,i2.

QUEST9. [Loop on j.] j <-- j + 1. If j .LE. 10, then go to step QUEST6.

QUEST10. [Update offset into common block.] offset <-- offset + txtwds.

QUEST11. [Loop on i.] i <-- i + 1. If i .LE. 5, then go to step QUEST5. Otherwise algorithm is complete.

QUEST12. [Present symbolic (numeric) queries.] Write out e1, e2.

ISMS SUBROUTINE/FUNCTION NAME - ELIM

FUNCTION - Eliminates an element from a given binary matrix.

USAGE - CALL ELIM(N,MATRIX,INDEX,DELETE,SYS,)

PARAMETERS

N - Input/Output integer scalar denoting the number of elements in the argument matrix.

MATRIX - Input/Output logical two dimensional matrix of dimensions "SYS" X "SYS". This is the matrix to be operated on.

INDEX - Input/Output integer vector of length "SYS". This is the index set of "MATRIX".

DELETE - Input integer scalar denoting the subscript of the row and column of the element to be eliminated.

SYS - Input integer scalar used in FORTRAN DIMENSION statements.

COMMON BLOCKS - None

REQUIRED ISMS ROUTINES - None

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm ELIM (Eliminate an element from a matrix). Given an argument matrix, M, its associated index set vector, R, the number of elements in M, n, and the row and column to eliminate, d, remove row and column d on M and shift the matrix up and to the left to get rid of blank row and column.

```
ELIM1.  [Initialize.] nminus <-- n - 1.

ELIM2.  [Check to see if row and column to be deleted is last logical position on M.] If d .EQ.  n, then go to step ELIM14.

ELIM3.  [Initialize row1.] row1 <-- d.

ELIM4.  [Initialize col.] row2 <-- row1 + 1, col <-- 1.

ELIM5.  [Move all columns below "d" over by 1.] m[row1,col] <-- m[row2,col].

ELIM6.  [Loop on col.] col <-- col + 1.  If col .LE.  n, then go to step ELIM5.

ELIM7.  [Initialize row.] row <-- 1.

ELIM8.  [Move all rows below "d" up by 1.] m[row,row1] <-- m[row,row2].

ELIM9.  [Loop on row.] row <-- row + 1.  If row .LE.  n, then go to step ELIM8.

ELIM10.  [Loop on row.] row1 <-- row1 + 1, then go to step ELIM4.

ELIM11.  [Initialize row1.] row1 <-- d.

ELIM12.  [Fix up index set.] row2 <-- row1 + 1, r[row1] <-- r[row2].

ELIM13.  [Loop on row1.] row1 <-- row1+ 1.  If row1 .LE.  nminus, then go to step ELIM12.

ELIM14.  [Set n to reflect deleted element.] n <-- n - 1.
```

ISMS SUBROUTINE/FUNCTION NAME - ADDEL

FUNCTION - Adds elements to a reachability matrix.

USAGE - CALL ADDEL(N,MATRIX,INDEX,TTYIN,TTYOUT,SYS)

PARAMETERS

N - Input/Output integer scalar indicating number of elements currently on "MATRIX".

MATRIX - Input/Output logical two dimensional matrix of dimensions "SYS" X "SYS". This is the binary reachability matrix that elements will be added to.

INDEX - Input/Output integer vector of length "SYS" containing the index set of "MATRIX".

TTYIN - Input integer scalar used as unit number in FORTRAN READ statements directed to interactive terminal.

TTYOUT - Input integer scalar used as unit number in FORTRAN WRITE statements directed to interactive terminal.

SYS - Input integer scalar used in FORTRAN DIMENSION statements.

COMMON BLOCKS - None

REQUIRED ISMS ROUTINES - GETNUM,FINDIT

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm ADDEL (Adds elements to a reachability matrix). Given the number of elements currently in a reachability matrix, n, the reachability matrix, R, and its associated index set vector, T, accept and add element numbers from interactive terminal.

ADDEL1. [Make sure memory limits are not exceeded.] If n .GE. SYS, then write error message and then algorithm is complete.

ADDEL2. [Accept element number to be added.] Apply Algorithm GETNUM to get n1. If n1 = 0, then algorithm is complete.

ADDEL3. [Make sure that n1 does not already exist in index set.] Apply Algorithm FINDIT. If n1 already exists, then write out error message and then go to step ADDEL2.

ADDEL4. [Add element.] n <-- n + 1, t[n] <-- n1, r[j,n] = 0, r[n,j] = 0, j=1,2,...,n. Set r[n,n] = 1 and then go to step ADDEL1.

ISMS SUBROUTINE/FUNCTION NAME - POOL

FUNCTION - Combines two elements on the same level or stage that are not connected.

USAGE - CALL POOL(N,REA,INDEX,TTYIN,TTYOUT,SYS)

PARAMETERS

N - Input/Output integer scalar equal to the number of elements in "REA".

REA - Input/Output logical two dimensional matrix of dimensions "SYS" X "SYS".

INDEX - Input/Output integer vector of length "SYS" that contains the index set of "REA".

TTYIN - Input integer scalar used as unit number in FORTRAN READ statements directed to interactive terminal.

TTYOUT - Input integer scalar used as unit number in FORTRAN WRITE statements directed to interactive terminal.

SYS - Input integer scalar used in FORTRAN DIMENSION statements.

COMMON BLOCKS - None

REQUIRED ISMS ROUTINES - GETNUM,FINDIT,HIERCH,COMBIN,TRNCLS

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm POOL (Combines two elements on the same level or stage). Given a reachability matrix, R, its associated index set vector, T, and the number of elements in R, n, ask the user for two elements and perform the pooling process.

```
POOL1.  [Accept two elements from interactive terminal.] Apply
        Algorithm GETNUM.

POOL.2 [Check if numbers are zero.] If u = 0 or v = 0, then algorithm
        is complete.

POOL3.  [Check if numbers are in index set.] Apply Algorithm FINDIT.
        If u and/or v are/is not found, then write appropriate error
        message and then go to step POOL1.

POOL4.  [Initialize.] stages <-- 0.

POOL5.  [Calculate the number of elements on each level of "REA".]
        Apply Algorithm HIERCH to fill up local vector L and H.

POOL6.  [Initialize.] f1 <-- 0, f2 <-- 0, start <-- 1, i <-- 1.

POOL7.  [Initialize.] end <-- start +l[i] - 1, j <-- start.

POOL8.  [Determine if u and v are on same level or stage.] If h[j] =
        u, then f1 <-- 1.  If h[j] = v, then f2 <-- 1.

POOL9.  [Loop on j.] j <-- j + 1.  If j .LE.  end, then go to step
        POOL8.

POOL10.  [Check if u and v are on same level or stage.] If f1 = 1 and
         f2 = 1, then go to step POOL19.

POOL11.  [Check if either u or v was found on this level or stage.] If
         f1 = 1 or f2 = 1, then go to step POOL13.

POOL12.  [Check next level.] start <-- end + 1, i <-- i + 1.  If i
         .LE.  number of levels or stages then go to step POOL7.

POOL13.  [Not on same level, see if on same stages.] If stages = 1,
         then go to step POOL18.

POOL14.  [Transpose R to obtain number of elements on each stage.] Set
         local matrix Z, z[i,j] <-- r[j,i], i=1,2,...,n, j=1,2,...,n.

POOL15.  [Copy transposed matrix into R.] r[i,j] <-- z[i,j], i=1,2,...
         ,n, j=1,2,...,n.

POOL16.  [Calculate number of elements on each stage of "REA".] Apply
         Algorithm HIERCH to fill up local vectors L and H.
```

POOL17. [Transpose again in order to get back original matrix.] r[i, j] <-- z[i,i], i=1,2,...,n, j=1,2,...,n. Set stages, stages <-- 0, then go to step POOL6.

POOL18. [Pooling error message, not on same level or stage.] Write appropriate error message to interactive terminal and then go to step POOL1.

POOL19. [Determine row and column subscripts for u and v on "REA".] Apply Algorithm FINDIT to get variables iu and iv.

POOL20. [Accept index set name for pooled elements from interactive terminal.] Apply Algorithm GETNUM to get newnam.

POOL21. [Check to see if newnam is zero.] If newnam = 0, then algorithm is complete.

POOL22. [Check to see if newnam is already in index set.] Apply Algorithm FINDIT. If newnam is already in index set, then write appropriate error message on interactive terminal and then go to step POOL20.

POOL23. [Combine elements u and v and use newnam as index in index set.] Apply Algorithm COMBIN.

POOL24. [Calculate reachability of new element.] Apply Algorithm TRNCLS and then go to step POOL1.

EREDGE

ISMS SUBROUTINE/FUNCTION NAME - EREDGE

FUNCTION - Erases an edge from the minimum edge digraph.

USAGE - CALL EREDGE(N,REA,INDEX,TTYIN,TTYOUT,SYS)

PARAMETERS

N - Input integer scalar indicating the number of elements currently in "REA".

REA - Input/Output logical two dimensional matrix of dimensions "SYS" X "SYS". This is the binary reachability matrix containing the minimum edge digraph.

INDEX - Input integer vector of length "SYS" containing the index set for "REA".

TTYIN - Input integer scalar used as unit number in FORTRAN READ statements directed to interactive terminal.

TTYOUT - Input integer scalar used as unit number in FORTRAN WRITE statements directed to interactive terminal.

SYS - Input integer scalar used in FORTRAN DIMENSION statements.

COMMON BLOCKS - None

REQUIRED ISMS ROUTINES - GETNUM,FINDIT,HIERCH,STAN,SWITCH,CONDE,ELIM,SKLTN,TRNCLS,IO

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm EREDGE (Erase an edge on the minimum edge digraph). Given the number of elements currently in a reachability matrix, n, the reachability matrix containing the minimum edge digraph, R, and its associated index set vector, T, accept edges to be eliminated from interactive user.

EREDGE1. [Accept edge to be eliminated.] Apply Algorithm GETNUM to get n1 and n2.

EREDGE2. [See if termination is requested.] If n1 = 0 or n2 = 0, then algorithm is complete.

EREDGE3. [Check to see if n1 and n2 are in system index set.] Apply Algorithm FINDIT. If n1 and/or n2 are/is not in system index set, then issue appropriate error message and then go to step EREDGE1.

EREDGE4. [Check to see if n1 and n2 are members of a cycle.] If r[ii,jj] = 1 and r[jj,ii] = 1, issue error message and then go to step EREDGE1.

EREDGE5. [Calculate non/redundant adjacency matrix for R.] Apply Algorithms HIERCH, STAN, CONDE, SKLTN.

EREDGE6. [Check to see if n1 and n2 are on minimum edge digraph.] Apply Algorithm FINDIT. If n1 and/or n2 are/is not on minimum edge digraph, then issue appropriate error message and then go to step EREDGE1.

EREDGE7. [Check to see if edge from ni to n2 exists on minimum edge digraph.] If r[ii,jj] = 0, then issue error message and then go to step EREDGE1.

EREDGE8. [Obtain index positions of n1 and n2 on R.] Apply Algorithm FINDIT.

EREDGE9. [Initialize.] l <-- 1.

EREDGE10. [Check to see if l is not a member of the antecedent set of n1.] If r[l,ii] = 0, then go to step EREDGE14.

EREDGE11. [Initialize.] k <-- 1.

EREDGE12. [Check to see if k is not a member of the reachability set of n2.] If r[jj,k] = 0, then go to step EREDGE13, else set r[l,k] <-- 0

EREDGE13. [Loop on k.] k <-- k + 1. If k .LE. n, then go to step EREDGE12.

EREDGE14. [Loop on l.] l <-- l + 1. If l .LE. n, then go to step EREDGE10.

EREDGE15. [Calculate reachability.] Apply Algorithm TRNCLS.

EREDGE16. [Update permanent file.] Apply Algorithm IO and then go to step EREDGE1.

ISMS SUBROUTINE/FUNCTION NAME - ADEDGE

FUNCTION - Adds edges on the minimum edge digraph.

USAGE - CALL ADEDGE(N,REA,INDEX,TTYIN,TTYOUT, SYS)

PARAMETERS

N - Input/Output integer scalar indicating the umber of elements currently on "REA".

REA - Input/Output logical two dimensional matrix of dimensions "SYS" X "SYS". This is the reachability matrix containing the minimum edge digraph.

INDEX - Input integer vector of length "SYS" containing the index set of "REA".

TTYIN - Input integer scalar used as unit number in FORTRAN READ statements directed to interactive terminal.

TTYOUT - Input integer scalar used as unit number in FORTRAN WRITE statements directed to interactive terminal.

SYS - Input integer scalar used in FORTRAN DIMENSION statements.

COMMON BLOCKS - None

REQUIRED ISMS ROUTINES - GETNUM,FINDIT,TRNCLS

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm ADEDGE (Add an edge on the minimum edge digraph). Given the number of elements currently in a reachability matrix, n, the reachability matrix containing the minimum edge digraph, R, and its associated index set, T, ask the interactive user for edges (relationships) to be added on the minimum edge digraph.

ADEDGE1. [Accept two element numbers from interactive terminal.] Apply Algorithm GETNUM to obtain n1 and n2.

ADEDGE2. [Check for termination directive.] If n1 = 0 or n2 = 0, then go to step ADEDGE5.

ADEDGE3. [Check to see if n1 and n2 are members of index set.] Apply Algorithm FINDIT. If n1 and/or n2 are/is not in index set, write error message on interactive terminal and then go to step ADEDGE1.

ADEDGE4. [Put edge in.] r[ii,jj] <-- 1, then go to step ADEDGE1.

ADEDGE5. [Transitively close matrix.] Apply Algorithm TRNCLS and then algorithm is complete.

ISMS SUBROUTINE/FUNCTION NAME - ERASE

FUNCTION - Erases elements from a binary matrix.

USAGE - CALL ERASE(N,MATRIX,INDEX,TTYIN,TTYOUT,SYS)

PARAMETERS

N - Input/Output integer scalar indicating the number of elements currently in "MATRIX".

MATRIX - Input/Output logical two dimensional matrix of dimensions "SYS" X "SYS". This is the argumet binary matrix.

IDEX - Input/Output integer vector of legth "SYS" containing the index set of "MATRIX".

TTYIN - Input integer scalar used as unit number in FORTRAN READ statements directed to interactive terminal.

TTYOUT - Input integer scalar used as unit number in FORTRAN WRITE statements directed to interactive terminal.

SYS - Input integer scalar used in FORTRAN DIMENSION statements.

COMMON BLOCKS - None

REQUIRED ISMS ROUTINES - ELIM,GETNUM,FINDIT

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm ERASE (Erase elements from a binary matrix). Given the number of elements currently in a binary matrix, n, the binary matrix, R, and its associated index set vector, T, accept and erase elements numbers from interactive terminal and erase them.

ERASE1. [Check to make sure n is not .LE. 0.] If n .LE. 0, then write out error message on interactive terminal and algorithm is complete.

ERASE2. [Accept an element number from interactive terminal.] Apply Algorithm GETNUM to obtain n1. If n1 = 0, then algorithm is complete.

ERASE3. [Check to see if n1 is a member of index set.] Apply Algorithm FINDIT. If n1 is not a member, issue appropriate error message and then go to step ERASE2.

ERASE4. [Erase element from matrix.] Apply Algorithm ELIM and then go to step ERASE2.

ISMS SUBROUTINE/FUNCTION NAME - TRNCLS

FUNCTION - Transitively closes a binary matrix to yield a reachability matrix.

USAGE - CALL TRNCLS(N,MATRIX,SYS)

PARAMETERS N - Input integer scalar indicating number of elements in "MATRIX".

MATRIX - Input/Output logical two dimensional matrix of dimensions "SYS" X "SYS". this is the argument binary matrix.

SYS - Input integer scalar used in FORTRAN DIMENSION statements.

COMMON BLOCKS - None

REQUIRED ISMS ROUTINES - None

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm TRNCLS (Transitive closure). Given the number of elements currently in a matrix, n, and the binary matrix, M, calculate and rearrange M into a reachability matrix.

TRNCLS1. [Initialize.] i <-- 1.

TRNCLS2. [Initialize element #I search.] nones <-- 0, last <-- 0.

TRNCLS3. [Initialize.] k <-- 1.

TRNCLS4. [Find all ones in reachability set of element #i.] If m[i,k] = 0, then go to step TRNCLS6.

TRNCLS5. [Found a one, keep track of it.] nones <-- nones + 1, fill local vector C, c[nones] <-- k.

TRNCLS6. [Loop on k.] k >-- k + 1. If k .LE. n, then go to step TRNCLS4.

TRNCLS7. [Check to see if any new ones were added from last time through.] If nones = last, then go to step TRNCLS14.

TRNCLS8. [No, compute new elements in reachability set of element #I by transitivity.] last <-- nones, l <-- 1.

TRNCLS9. [Initialize.] k <-- c[l], j <-- 1.

TRNCLS10. [Fill up matrix.] If m[i,k] = 1 and m[k,j] = 1, then set m[i,j] <-- 1.

TRNCLS11. [Loop on j.] j <-- j + 1. If j .LE. n, then go to step TRNCLS10.

TRNCLS12. [Loop on l.] l <-- l + 1. If l .LE. nones, then go to step TRNCLS9.

TRNCLS13. [Continue processing.] nones <-- 0, then go to step TRNCLS3.

TRNCLS14. [Loop on i.] i <-- i + 1. If i .LE. n, then go to step TRNCLS2, else algorithm is complete.

ISMS SUBROUTINE/FUNCTION NAME - PACK

FUNCTION - Packs as many display characters as possible into one machine word.

USAGE - CALL PACK(NWORDS,NCHAR,CARD)

PARAMETERS

NWORDS - Input integer scalar equal to the number of machine words required to hold 600 characters plus 10 additional words.

NCHAR - Input integer scalar equal to the number of characters able to be stored in one machine word.

CARD - Input integer vector of length 60 containing the text to be packed, stored one character per word.

COMMON BLOCKS - Blank common is utilized.

COMMON PARAMETERS - See the description for subroutine MAKEIT.

REQUIRED ISMS ROUTINES - None

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm PK (Packing Algorithm). Given a maximum of 60 characters stored in one word, pack as many characters as possible into one word and store the packed string starting on the first available word boundary.

PK1. [Initialize.] pos <-- pos + 1, i <-- 1

PK2. [Compute number of characters to process.] If card(61 - i) is not a blank, then go to step PK4.

PK3. [Loop on i.] i <-- i + 1. If i < 61, then go to step PK2, else i = 59.

PK4. [Calculate actual length.] len <-- 61 - i

PK5. [Encode text (i.e., PACK).] length(pos) <-- (len - 1)/nchar + 1, end <-- start + length(pos). Store reformatted text into "packed", packed(i), i=start,end. start <-- start + length(pos) and algorithm is complete.

ISMS SUBROUTINE/FUNCTION NAME - WRITE

FUNCTION - Writes a record onto the direct access query file.

USAGE - CALL WRITE(LOGPOS,NWORDS)

PARAMETERS
LOGPOS - Input integer scalar denoting the logical position of the record in the file.
NWORDS - Input integer scalar equal to the number of machine words required to hold 600 characters plus 10 additional words.

COMMON BLOCKS - Blank common is utilized.

COMMON PARAMETERS - See the description for subroutine MAKEIT.

REQUIRED ISMS ROUTINES - None

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm WR (Write Algorithm). Given a record of size "nwords", write the record onto the query file in the proper position.

WR1. [Determine if an element text is to be written.] If logpos < 5, then go to step WR3.

WR2. [An element text is to be written.] n <-- n + 1

WR3. [Write record.] Write record onto file in position "logpos".

WR4. [Reset record associated variables.] start <-- 1, pos <-- 0, length(i) <-- 0, i=1,2,...,10 and algorithm is complete.

MAKEIT

ISMS SUBROUTINE/FUNCTION NAME - MAKEIT

FUNCTION - Converts a sequential access file containing queries for an ISM session into a direct access file for random retrieval.

USAGE - CALL MAKEIT

PARAMETERS - None

COMMON BLOCKS - Blank common is utilized.

COMMON PARAMETERS

- TTYIN - Integer scalar used as unit number in FORTRAN READ statements directed to interactive terminal.
- TTYOUT - Integer scalar used as unit number in FORTRAN WRITE statements directed to interactive terminal.
- N - Integer scalar equal to the number of element text records processed.
- TOTAL - Integer scalar equal to the number of machine words required for the current element text record.
- POS - Integer scalar which denotes the line number (1 thru 10) of the current element text record being processed.
- START - Integer scalar which is equal to the first available subscript of "PACKED" for packing in the current element text record.
- LENGTH - Integer vector of length 10 words. LENGTH(I) = number of machine words required to store line#I of the element text record currently being processed.
- PACKED - Integer vector of length 200 words containing the text of the element text record with each line stored on a word boundary.

REQUIRED ISMS ROUTINES - DOIT,SHOW,PACK,WRITE,GETNUM

REQUIRED FORTRAN ROUTINES - ENCODE (if available)

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm MK (Makeit). Given a line oriented sequential text file with textual and control information, reformat and write the information to a direct access file.

MK1. [Initialize.] nchar <-- number of bits per word/number of bits required to represent one character, nwords <-- (60/nchar)*10

MK2. [Construct file.] Apply Algorithm DOIT.

MK3. [Write number of records on first record.] Write, variable n.

MK4. [Ask user if he wants to show elements.] Write message and read response. If user does not want to see his elements then go to step MK6.

MK5. [Show elements.] Apply Algorithm SHOW.

MK6. [Terminate.] Algorithm is complete.

ISMS SUBROUTINE/FUNCTION NAME - DOIT

FUNCTION - Creates the random text file.

USAGE - CALL DOIT(NCHAR,NWORDS,CARD)

PARAMETERS

NCHAR - Input integer scalar equal to the number of characters able to be stored in one machine word.

NWORDS - Input integer scalar equal to the number of machine words required to hold 600 characters plus 10 additional words.

CARD - Input integer vector of length 60 words.

COMMON BLOCKS - Blank common is utilized.

COMMON PARAMETERS - See the description for subroutine MAKEIT.

REQUIRED ISMS ROUTINES - None

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm DO (Doit Algorithm). Given a sequential text file with text and control cards, read each record and form a random access text file for full text queries.

DO1. [Initialize.] start <-- 1, flag <-- 0, n <-- 0, length(i) <-- 0, i=1,2,...,10.

DO2. [Read a record from text file.] Read in sixty characters into "card" with each character on a word boundary. On end of file, algorithm is complete.

DO3. [Check for control card.] If CARD(1) = "/" then go to step DO5.

DO4. [Pack text.] The record read was a text record so apply Algorithm PK and then go to step DO2.

D05. [Check to see if current record should be written.] If flag = 0 then go to step D07.

D06. [Write current record to random text file.] Apply Algorithm WRITE.

D07. [Parse control card.] logpos <-- 0. If CARD(2) not equal to an "R" then go to step D010.

D08. [Determine logical position for next record.] If CARD(3) = i, then logpos <-- i + 1, i=1,2,3.

D09. [Check for error on control card.] If logpos < 2, then go to step D013, else flag = 1 and go to step D02.

D010. [Continue parsing control card.] If CARD(2) is not equal to an "E", then go to step D012.

D011. [Continue parsing control card.] If CARD(3) = "L", then logpos <-- n + 5. If logpos is less than 2, then go to step D013, else set flag = 1 and go to step D02.

D012. [Check for end of file.] If CARD(2) = "/", then algorithm is complete.

D013. [Write error on interactive terminal.] Write "control card error" and then algorithm is complete.

ISMS SUBROUTINE/FUNCTION NAME - SHOW

FUNCTION - Displays all or user specified pairs of elements as they might appear during an ISM session.

USAGE - CALL SHOW

COMMON BLOCKS SHOWB - Named integer common block of length 5*"NWORDS."
- blank common is also utilized. See the description for subroutine MAKEIT.

COMMON PARAMETERS - See the description for subroutine BORDER for the list of parameters.
- SHOWB has been made a common block to guarantee the contingincy of storage.

REQUIRED ISMS ROUTINES - None

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm SH (Show Algorithm). Given a direct access query file, display user specified sets of queries.

SH1. [Initialize.] alls <-- 0, el1 <-- 0, el2 <-- 0

SH2. [Read in relational clauses 1,2, and 3.] Read R1, R2, R3.

SH3. [See if user wants to display all elements.] Ask user, if yes, then go to step SH13.

SH4. [Ask user which two elements to present.] Ask user, if one or two zeros are typed, then algorithm is complete. Otherwise, make sure that elements text exist. If not, then issue an error message and then go to step SH4.

SH5. [Read in elements text.] i1 <-- el1 + 4, i2 <-- el2 + 4, read in records i1 and i2.

SH6. [Initialize for typing out.] offset <-- 0, i <-- 1

SH7. [Initialize for ten phrases.] j <-- 1, i1 <-- 0, i2 <-- 0

SH8. [Check length indicator.] If SHOWB(j + offset) = 0, then go to step SH10.

SH9. [Print out this phrase.] length <-- SHOWB(J + OFFSET), I1 <-- I2 + 1, I2 <-- I1 + LENGTH + 1, WRITE OUT SHOWB(c + offset + 10), c=i1,...,i2.

SH10. [Loop on j.] j <-- j + 1. If j < 11, then go to step SH8.

SH11. [Loop on i.] offset <-- offset + nwords, i <-- i + 1. If i < 6, then go to step SH7.

SH12. [Check for desired printing of all element pairs.] If alls = 0, then go to step SH4.

SH13. [Come here when printing all element pairs.] el1 <-- el2 + 1, el2 <-- el2 + 1. If el1 > n, then algorithm is complete, if el2 > n, then set el2 <-- 1, alls = 1, then go to step SH5.

ISMS SUBROUTINE/FUNCTION NAME		ADD
FUNCTION		Adds elements to the weighted matrix.
USAGE		CALL ADD(N,MAT,INDEX)
PARAMETERS	N	Input/Output integer scalar indicating the current number of elements in the weighted matrix.
	MAT	Input/Output integer two dimensional matrix of dimensions 50 X 50. This is the weighted matrix.
	INDEX	Input/Output integer vector of length 50 words containing the index set for "MAT".
COMMON BLOCKS	INFO	Named integer common block 3 words long.
COMMON PARAMETERS		See the description for subroutine PRNTMT.
REQUIRED ISMS ROUTINES		FINDIT,GETNUM,QUEST
REQUIRED FORTRAN ROUTINES		None
ALGORITHM EMPLOYED		The algorithm employed is described below:

Algorithm ADD (Add an element to a weighted matrix). Given the number of elements currently in a weighted matrix, n, the weighted matrix, W, and its associated index set, T, add an element to the weighted matrix.

ADD1. [Ask user for new element to be added.] Prompt user and then apply algorithm GETNUM.

ADD2. [Make sure that number is less than or equal to 9999 for fortran formatting.] If number read in is > 9999, issue error message and then go to step ADD1.

ADD3. [Check for a zero input.] If number typed was a zero, then algorithm is complete.

ADD4. [Check for duplicate element.] Apply Algorithm FINDIT. If element is already in index set, the issue error message and then go to step ADD1.

ADD5. [Put new element into matrix.] n <-- n + 1, t[n] <-- new number.

ADD6. [See if user wants to fill up relationships for new element.] Prompt user. If user types an "N", then go to step ADD1.

ADD7. [Initialize column fill.] i <-- 1

ADD8. [Present question.] Apply Algorithm QUEST(t[t],t[n]).

ADD9. [Get weight value.] Apply Algorithm GETNUM.

ADD10. [Make sure weight value is less than 10.] If weight value is greater then 9, issue error message and then go to step ADD8.

ADD11. [Set matrix.] w[i,n] <-- weight value

ADD12. [Loop on i.] i <-- i + 1. If i is less than or equal to n - 1, then go to step ADD8.

ADD13. [Initialize row fill.] i <-- 1

ADD14. [Present question.] Apply Algorithm QUEST(t[n],t[i])

ADD15. [Get weight value.] Apply Algorithm GETNUM.

ADD16. [Make sure weight value is less than 10.] If weight value is greater than 9, issue error message and then go to step ADD14.

ADD17. [Set matrix.] w[n,i] <-- weight value

ADD18. [Loop on i.] i <-- i + 1. If i is less than or equal to n - 1, then go to step ADD14, else go to step ADD1.

PRNTMT

ISMS SUBROUTINE/FUNCTION NAME - PRNTMT

FUNCTION - Prints out all weighted relationships included in a weighted matrix that are greater than or equal to an input weight value.

USAGE - CALL PRNTMT(N,MAT,INDEX,THRESH,SELPNT)

PARAMETERS

N - Input integer scalar equal to the number of elements on "MAT".

MAT - Input integer two dimensional matrix of dimensions 50 X 50. This is the weighted matrix.

INDEX - Input integer vector of length 50 containing the index set for "MAT".

THRESH - Input integer scalar used as cut-off threshold. All relationships greater than or equal to "THRESH" will be printed if "SELPNT" = .TRUE..

SELPNT - Input logical variable used to determine if only relationships greater than or equal to "THRESH" should be printed (SELPNT = .TRUE.) or all relationships (SELPNT = .FALSE.)

COMMON BLOCKS

INFO - Named integer common block 3 words long.

COMMON PARAMETERS

QTYPE - Input logical variable indicating the type of query to be presented.
QTYPE = .FALSE. causes full text queries to be printed.
QTYPE = .TRUE. causes symbolic queries to be printed.

TTYIN - Input integer scalar used as unit number in FORTRAN READ statements directed to interactive terminal.

TTYOUT - Input integer scalar used as unit number in in FORTRAN WRITE statements directed to interactive terminal.

REQUIRED ISMS ROUTINES - None

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm PT (Print weighted matrix). Given the number of elements in a weighted matrix, n, the weighted matrix, W, and its associated index set vector, M, print out all or just relationships greater than or equal to a threshold, t.

PT1. [Initialize.] If selpnt = 1, then print special title. Set i, i <-- 1.

PT2. [Begin row search.] Set ctr <-- 0, j <-- 1.

PT3. [Check for printing greater than or equal to threshold.] If selpnt = 0, then go to step PT5.

PT4. [Check for relationships greater than or equal to the threshold.] if w[i,j] < t, then go to step PT7.

PT5. [Don't print diagonal.] If i = j, then go to step PT7.

PT6. [Store values in local array for printing.] ctr <-- ctr + 1, list[ctr,1] <-- m[j], list[ctr,2] <-- w[i,j].

PT7. [Loop on j.] j <-- j + 1. If j <= n, then go to step PT3.

PT8. [Check for ctr=0.] If ctr = 0, then write out m[i] and then go to step PT10.

PT9. [Write out row relationships.] Write out m[i], list[k,1], list[k,2], k=1,2,....,ctr.

PT10. [Loop on i.] i <-- i + 1. If i <= n, then go to step PT2, else algorithm is complete.

CHANGE

ISMS SUBROUTINE/FUNCTION NAME - CHANGE

FUNCTION - Changes weights in the weighted matrix via user interaction.

USAGE - CALL CHANGE(N,MAT,INDEX)

PARAMETERS

N - Input integer scalar indicating the current number of elements in "MAT".

MAT - Input/Output integer two dimensional matrix of dimensions 50 X 50 containing the weighted matrix.

INDEX - Input integer vector of length 50 words containing the index set for "MAT".

COMMON BLOCKS

INFO - Named integer common block of length 3 words.

COMMON PARAMETERS - See the description for subroutine PRNTMT.

REQUIRED ISMS ROUTINES - FINDIT,GETNUM

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm CH (Change Algorithm). Given the number of elements currently in a weighted matrix, n, the weighted matrix, W, and its associated index set, T, allow changes in the matrix via user interaction.

CH1. [Prompt user.] Write message to interactive terminal.

CH2. [Read change.] Apply Algorithm GETNUM. Read the element numbers and weight change (3 numbers).

CH3. [Check for termination.] If any of the element numbers are zero, algorithm is complete.

CH4. [Check to see if element numbers are valid.] Apply Algorithm FINDIT. If one or both numbers are invalid, write an error message and then go to step CH1.

CH5. [Make sure weight typed is 9 or less.] If weight value is greater than 9, write error message and then go to step CH1.

CH6. [Change weight.] w[x,y] <-- weight, where x and y are outputs of algorithm FINDIT. Go to step CH1.

ISMS SUBROUTINE/FUNCTION NAME - FILL

FUNCTION - Helps the user fill the weighted matrix by presenting the required queries.

USAGE - CALL FILL(N,MAT,INDEX,R1,R2)

PARAMETERS
N - Input integer scalar indicating the current number of elements in "MAT".
MAT - Input/Output integer two dimensional matrix of dimensions 50 X 50p. This is the weighted matrix.
INDEX - Input integer vector of length 50 words contains the index set for "MAT".
R1 - Input/Output integer scalar used as the row restart indice.
R2 - Input/Output integer scalar used as the column restart indice.

COMMON BLOCKS
INFO - Named integer common block of length 3 words.

COMMON PARAMETERS - See the description for subroutine PRNTMT.

REQUIRED ISMS ROUTINES - QUEST,GETNUM

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is desribed below:

Algorithm FL (Fill Algorithm). Given the number of elements currently in a weighted matrix, n, the weighted matrix, W, its associated index set, T, and two restart parameters r1 and r2, allow the filling of all positions (with the exception to the diagonal) on the weighted matrix.

FL1. [Check to see if this is a restart.] If r1 or r2 is greater than zero, go to step FL12.

FL2. [Initialize.] row <-- 1, col <-- 1, r1 <-- 0, r2 <-- 0

FL3. [Initialize row loop.] i <-- row

FL4. [Initialize column loop.] j <-- col

FL5. [Don't process diagonal.] If i = j, then go to step FL10.

FL6. [Present query.] Apply Algorithm QUEST(t[i],t[j]).

FL7. [Obtain weight.] Apply Algorithm GETNUM.

FL8. [Check for valid weight input.] If weight is equal to 10, then go to step FL19. If weight is greater than 9, then issue error message and go to step FL6.

FL9. [Set matrix.] w[i,j] <-- weight value

FL10. [Loop on j.] j <?s- j + 1. If j is greater than or equal to n, then go to step FL5.

FL11. [Loop on i.] i <-- i + 1. If i is less than or equal to n, then go to step FL4, else algorithm is complete.

FL12. [Restart questioning.] i <-- r1, j <-- r2

FL13. [Present query.] Apply Algorithm QUEST(t[i],t[j]).

FL14. [Obtain weight.] Apply Algorithm GETNUM.

FL15. [Check for valid weight input.] If weight is equal to 10, then go to step FL19. If weight is greater than 9, then issue error message and go to step FL13.

FL16. [Set matrix.] w[i,j] <-- weight value

FL17. [Loop on j.] j <-- j + 1. If j is less than or equal to n, then go to step FL13.

FL18. [Reset restart parameters.] row <-- 1, col <-- r1 + 1, r1 <-- 0, r2 <-- 0, then go to step FL3.

FL19. [Generate restart parameters.] r1 <-- i, r2 <-- j, then algorithm is complete.

ISMS SUBROUTINE/FUNCTION NAME - RESOLV

FUNCTION - Resolves the threshold of a weighted matrix.

USAGE - CALL RESOLV(N,MAT,INDEX,THRESH)

PARAMETERS N - Input integer scalar indicating the current number of elements in "MAT".

MAT - Input integer two dimensional matrix of dimensions 50 X 50. This is the weighted matrix.

INDEX - Input integer vector of length 50 words containing the index set for "MAT".

THRESH - Input integer scalar equal to the maximum threshold to be used (usually equal to 9).

COMMON BLOCKS INFO - Named integer common block of length 3 words.

COMMON PARAMETERS - See the description for subroutine PRNTMT.

REQUIRED ISMS ROUTINES - TRNCLS,PRNTMT,GEOD,NOTR

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm RS (Resolve Algorithm). Given the number of elements currently in a weighted matrix, n, the weighted matrix, W, its associated index set, T, and the maximum threshold, r, resolve the maximum threshold of the matrix.

RS1. [Initialize.] z <-- r

RS2. [Initialize.] i <-- 1

RS3. [Begin constructing binary adjacency matrix.] j <-- 1

RS4. [Don't process main diagonal.] Set a[i,j] <-- 0. If i = j, then go to step RS6.

RS5. [Check threshold.] If w[i,j] is less than z, then go to step RS7.

RS6. [Set binary matrix position.] a[i,j] <-- 1

RS7. [Loop on j.] j <-- j + 1. If j is less than or equal to n, then go to step RS4.

RS8. [Loop on i.] i <-- i + 1. If i is less than or equal to n, then go to step RS3.

RS9. [Transitively close matrix.] Apply Algorithm TRNCLS to get reachability matrix b.

RS10. [Is binary matrix all ones ?] If b[i,j] = 0, then go to step RS14, j=1,2,...,n, i=1,2,...n.

RS11. [Tell user cycle is resolved.] Write z.

RS12. [Print universal matrix.] Apply Algorithm PRNTMT.

RS13. [Print geodetic paths.] Apply Algorithm GEOD and then algorithm is complete.

RS14. [Select next lowest threshold.] z <-- z - 1, then go to step RS2.

ISMS SUBROUTINE/FUNCTION NAME		- GEOD
FUNCTION		- Outputs the geodetic cycle paths contained within a weighted matrix.
USAGE		- CALL GEOD(ADJ,N,INDEX)
PARAMETERS	ADJ	- Input logical two dimensional matrix of dimensions 50 X 50. This is the binary threshold adjacency matrix formed in subroutine RESOLV.
	N	- Input integer scalar indicating the number of elements in "ADJ".
	INDEX	- Input integer vector of length 50 containing the index set for "ADJ".
COMMON BLOCKS	INFO	- Named integer common block of length 3 words.
COMMON PARAMETERS		- See the description for subroutine PRNTMT.
ALGORITHM EMPLOYED		- The algorithm employed is described below:

Algorithm GE (Geodetic Algorithm). Given the number of elements currently in a binary threshold matrix, n, the binary threshold matrix, A, and its associated index set vector, T, determine and print all geodetic cycle sets.

GE1. [Initialize.] i <-- 1, nbn <-- 1

GE2. [Initialize G1 formation.] j <-- 1

GE3. [Check for a zero.] If a[i,j] = 0, then go to step GE6.

GE4. [Set G and B.] g[i,j] <-- 1, b[i,j] <-- 1

GE5. [Subtract identity.] If i = j, then g[i,j] = 0.

GE6. [Loop on j.] j <-- j + 1. If j is less than or equal to n, then go to step GE3.

GE7. [Loop on i.] i <-- i + 1. If i is less than or equal to n, then go to step GE2.

GE8. [Initialize.] i <-- 1

GE9. [Initialize.] j <-- 1

GE10. [Zero C.] c(j,i) <-- 0

GE11. [Initialize.] k <-- 1

GE12. [Set switch.] ma <-- 0. If a[i,j] = 0, then ma <-- 1.

GE13. [Compute A**nbn-1.] c[j,i] <-- ma * b[k,i] + c[j,i]

GE14. [Loop on k.] k <-- k + 1. If k is less than or equal to n, then go to step GE12.

GE15. [Check for invalid entry.] If c[j,i] is not equal to 0, then set c[j,i] <-- 1.

GE16. [Loop on j.] j <-- j + 1. If j is less than or equal to n, then go to step GE10.

GE17. [Loop on i.] i <-- i + 1. If i is less than or equal to n, then go to step GE9.

GE18. [Initialize for G**nbn.] nbn <-- nbn + 1, i <-- 1

GE19. [Initialize.] j <-- 1

GE20. [Calculate G**nbn.] g[i,j] <-- g[i,j] + nbn * (c[i,j] - b[i,j]), b[i,j] <-- c[i,j].

GE21. [Loop on j.] j <-- j + 1. If j is less than or equal to n, then go to step GE20.

GE22. [Loop on i.] i <-- i + 1. If i is less than or equal to n, then go to step GE19.

GE23. [Is G formed yet ?] If nbn is less than n - 1, then go to step GE8.

GE24. [Initialize path computation.] i <-- 2, ncpt <-- 0, write heading on terminal.

GE25. [Set limit for j.] lim <-- i - 1, j <-- 1

GE26. [Find cycle distance.] nb <-- g[i,j], na <-- g[j,i]

GE27. [If distance is zero, don't process.] If na or nb = 0, then go to step GE47.

GE28. [Keep account of paths.] nbnd <-- 1, l[nbnd] <-- i

GE29. [Is path longer than 1 ?] If nb is greater than 1, then go to step GE31.

GE30. [Store end link.] nbnd <-- nbnd + 1, l[nbnd] <-- j, then go to step GE36.

GE31. [Initialize search for last link.] lmm <-- nb - 1, last <-- 0, in <-- 1

GE32. [Find link back to i.] nbnd <-- nbnd + 1, l[nbnd] <-- next link (Apply Algorithm NOTR), last <-- l[nbnd].

GE33. [Unsuccessful ?] If ix = 1, then write out error message and go to step GE36.

GE34. [Loop on in.] in <-- in + 1. If in is less than or equal to lmm, then go to step GE32.

GE35. [Store end link.] nbnd <-- nbnd + 1, l[nbnd] <-- j

GE36. [Is path complete ?] If na is greater than 1, then go to step GE38.

GE37. [Store return link.] l[nbnd + 1] <-- i, then go to step GE43.

GE38. [Initialize search for last link.] lmn <-- na - 1, last <-- 0, in <-- 1

GE39. [Find link back to j.] nbnd <-- nbnd + 1, l[nbnd] <-- next link (Apply Algorithm NOTR), last <-- l[nbnd].

GE40. [Unsuccessful ?] If ix = 1, then write out error message and go to step GE43.

GE41. [Loop on in.] in <-- in + 1. If in is less than or equal to lmn, then go to step GE39.

GE42. [Store end link.] l[nbnd + 1] <-- i

GE43. [Initialize for printout.] ncpt <-- ncpt + 1, lmc <-- nbnd + 1, nx <-- 1

GE44. [Fix up print vector for index set used.] num <-- l[nx], p[nx] <-- t[num]

GE45. [Loop on nx.] nx <-- nx + 1. If nx is less than or equal to lmc, then go to step GE44.

GE46. [Print out cycle path on terminal.] nxx <-- na + nb, write ncpt, nxx, t[i], t[j], p[k], k=1,2,...,lmc.

GE47. [Loop on j.] j <-- j + 1. If j is less than or equal to lim, then go to step GE26.

GE48. [Loop on i.] i <-- i + 1. If i is less than or equal to n, then go to step GE25, else algorithm is complete.

ISMS SUBROUTINE/FUNCTION NAME - NOTR

FUNCTION - Returns the next element in the geodetic path.

USAGE - X = NOTR(G,ICL,NBLG,ILG,NBCL,IX,NBN,LAST)

PARAMETERS

	G	- Input integer two dimensional matrix of dimensions 50 X 50. This is the distance matrix.
	ICL	- Input integer scalar equal to the starting search index.
	NBLG	- Input integer scalar equal to the distance of the link.
	ILG	- Input integer scalar equal to the ending search index.
	NBCL	- Input integer scalar equal to the distance of the desired link from the starting index.
	IX	- Output integer scalar used as an error switch for incomplete paths. IX = 0 means complete path. IX = 1 means no link found.
	LAST	- Input integer scalar equal to the last link found. This is required to keep the algorithm searching down the same path.

COMMON BLOCKS - None

REQUIRED ISMS ROUTINES - None

REQUIRED FORTRAN ROUTINES - None

ALGORITHM EMPLOYED - The algorithm employed is described below:

Algorithm NOTR. Given the number of paths in a distance matrix, n, the distance matrix, G, and other infor mation, determine the next element in the geodetic cycle.

NOTR1. [Initialize.] ix <-- 0, ibi <-- 0, i <-- 1

NOTR2. [Find all elements with distance nblg from ilg.] Ifg[i,ilg] is not equal to nblg, then go to step NOTR4.

NOTR3. [Keep track of elements with same distance.] ibi <-- ibi + 1, l[ibi] <-- i

NOTR4. [Loop on i.] i <-- i + 1. If i is less than or equal to n, then go to step NOTR2.

NOTR5. [If none found, than algorithm is complete.] If ibi = 0, then algorithm is complete.

NOTR6. [Initialize.] i <-- 1

NOTR7. [Search for an element of proper distance.] If g[icl,i] is not equal to nbcl, then go to step NOTR13.

NOTR8. [See if this is the first link.] If last = 0, then go to step NOTR10.

NOTR9. [Keep going on same path.] If g[last,i] is not equal to 1, then go to step NOTR13.

NOTR10. [Initialize.] k <-- 1

NOTR11. [See if this i is of proper distance.] If i = l[k], then go to step NOTR15.

NOTR12. [Loop on k.] k <-- k + 1. If k is less than or equal to ibi, then go to step NOTR11.

NOTR13. [Loop on i.] i <-- i + 1. If i is less than or equal to n, then go to step NOTR7.

NOTR14. [No link found.] ix <-- 1, then algorithm is complete.

NOTR15. [Return link.] notr <-- l[k], then algorithm is complete.

Attachment No. 2

INTERPRETIVE STRUCTURAL MODELING SOFTWARE

Prepared by:

David R. Yingling, Jr.

This User's Manual was developed for the ISM software on the University of Dayton computer. It illustrates the type of instructions that should be given to users wishing to utilize the ISM software.

PREFACE

This manual* is intended to serve as a user's guide for the Interpretive Structural Modeling Software package developed by the Engineering and Public Policy Group at the University of Dayton, and as implemented on UD's Univac Series 70 computer operating under VS/9. This version of the manual was written under the assumption that its user is familiar with the ISM** methodology and supportive concepts but has had little or no previous experience on a computer, specifically the UD computer. Therefore, all basic procedures like LOGging ON, etc. have been included so as to make this manual a stand alone guide.

It is suggested that if the user plans to make extensive use of text files that he or she obtain Manual #4.1 (EDT) from the University of Dayton Office for Computing Activities (OCA). This manual is available free of charge and will make using the Univac File Editor (EDT) and text file editing much easier.

If you find any discrepancies in this manual or ISMS-UD, please write to the address on the front page or call Monday thru Friday 0800 to 1500 Eastern time. If the telephone is busy or no answer, "leave word" at and we will return your call.

---

*The table of contents may be found at the end.

**Interpretive Structural Modeling (ISM) is developed in John N. Warfield's Societal Systems: Planning, Policy, and Complexity, Wiley-Interscience: New York, 1976.

INTRODUCTION

The University of Dayton ISMS Version 2.0, as implemented on UD's time sharing computer system, consists of three FORTRAN IV programs which perform embedding and amending, element text processing, and cycle resolution. The following is a brief description of each program.

ISMS-UD - is the program that provides embedding and amending facilities for Interpretive Structural Models. ISMS-UD maintains a reachability matrix permanent file. A query file supplies the information necessary for full text queries.

CYCLE - is the program that resolves the cycles contained in an Interpretive Structural Model. CYCLE creates and maintains a matrix containing numeric weights denoting the strength of relationships between elements of a cycle. A query file supplies the information necessary for full text queries.

MAKEII - is a program that restructures an EDT-created text file into a random access format required by the ISM method. The random access format is written onto the query file.

UD_TIME_SHARING_COMPUTER_SYSTEM

SYSTEM_HARDWARE

The UD Computing Facility owns a Univac Series 70/7 Time-Sharing Computer System. This third generation virtual memory multiprogramming system has 750K bytes of main core storage with a backing store of 6.4 million bytes. Time-sharing input/output to the processor is handled by a communications controller attached to one of the multiplexor channels. The communications controller presently allows for 18 remote harduire terminals and 33 remote dial-up terminals. Ten of the 33 dial-up ports are 300 baud with the remaining ports 110 baud.

SYSTEM_SOFTWARE

The Univac Series 70/7 computer presently runs under the operating system VS/9 Version 3.5. VS/9 is a group of programs and subprograms which control input, compilation, assembly, loading, execution, and output of all programs submitted to the computer as well as the allocation of system resources.

UNIVAC_FILE_EDITOR_(EDT)

If the user desires to have programs ISMS-UD and CYCLE present queries in an English text format, a two step procedure must be followed. Step 1: a sequential element text file must be created (using EDT). Step 2: the sequential element text file must then be converted into a query file (using MAKEIT).

EDT is a program that permits the creation and modification of sequential element text files. EDT is invoked under VS/9 and responds to simple commands prefixed with an @ ("at sign") which initiate, maintain, correct, and complete file construction. Although EDT is a comprehensive editing package capable of performing varied tasks, it presents minimal concern to the ISMS-UD user. A sample run using EDT can be seen in the Appendix of this document. For a more comprehensive discussion of EDT, the user is referred to OCA Manual #4.1 (EDT) available free of charge at the UD data center, Miriam Hall.

PROCEDURES_AND_IMPORTANT_SYSTEM_CONCEPTS

SCHEDULING THE USE OF THE UD COMPUTER

The Univac Series 70/7 is a multiprogramming computer system; that is, it is servicing many users all at the same time. Consequently, the more people using the system, the slower the turnaround time (response time). It is suggested that the user not schedule the use of the computer during prime time (i.e., 1 P.M. to 5 P.M. weekdays) and near the end of the UD academic terms (usually first two weeks in both December and April). It has also been experienced near the end of the terms that dial-up ports are extremely hard to get, so these words to the wise --PLAN AHEAD-- START EARLY!!!

AUTOMATIC LOGOFF FEATURE

In order to keep dial-up ports from being tied up or occupied by an inactive terminal user, the UD computer has attached to it a device which will AUTOMATICALLY LOGOFF A TERMINAL INACTIVE FOR NINE MINUTES. This is important to remember because sometimes the queries presented by ISMS-UD require a long period of time for thought and discussion. During this discussion time, the computer under control of ISMS-UD is waiting for an answer to the query; that is, the terminal is inactive. If an input/output is not done within nine minutes after the start of the read, the terminal is automatically logged off and the ISM is partially lost. It is suggested that the terminal manager or user keep an eye on the time and, if the nine minute limit draws near while discussion continues, type an invalid input to ISMS-UD. ISMS-UD will inform the user of the invalid input and re-prompt with the same query. The nine minute timer is then reset and the ISM saved.

SPECIAL CONTROL KEYS

End_of_Transmission_Key

Anyone who has used an interactive computer terminal knows that some special key on the terminal is used to signal the computer to take action on the typed input. On most computers, this is the RETURN key. The UD computer does_not_use_the_RETURN_key, but instead, the depressing of two keys simultaneously. These two keys are the CTRL and C keys, denoted as CTRL C. That is, when the user wants to type a command to the computer, he or she first types the command and then sends the input to the computer by holding down the CTRL key and typing a "C". It will be assumed from here on that the user understands this principle.

Error_Correcting_Keys

Two input error correcting keys are supported by the Univac hardware. They are:

"CTRL X" -- cancels the input line. The processor responds with %CNCL to indicate the line was canceled.

"SHIFT O" or "UNDERLINE" -- is essentially a backspace key. When typed, it tells the processor to ignore the last typed character before the SHIFT O or UNDERLINE. (The type of terminal used dictates which is appropriate.)

Both error correcting keys may be used while running the operating system and ISMS-UD.

PARITY AND DUPLEX SETTINGS FOR DIAL-UP OPERATION

The parity setting on the terminal must be EVEN.

The duplex setting on the acoustic coupler (and terminal if applicable) must be HALF.

PERTINENT TELEPHONE NUMBERS

The telephone numbers below should be kept handy.

- 110 BAUD
- 110 BAUD
- 300 BAUD

- SYSTEM STATUS/SCHEDULE (recording)
- HELP!!!!

HOW_TO_LOGON

There are two types of terminals that can be used on the UD computer: the hardwired terminal which is directly connected to the computer, and the acoustically coupled terminal which is connected to the computer via the telephone system. Each type of terminal requires a different LOGON procedure which will subsequently be discussed.

PROCEDURE FOR LOGGING ON A HARDWIRED TERMINAL

1. Turn on the power switch on the terminal.
2. Set the DUPLEX switch to HALF.
3. Set the LOCAL/LINE switch to LINE.

Note: On a Teletype Model 33, turn the control knob (below and to the right of the keyboard) counterclockwise instead of the above three steps.

4. While depressing the CTRL key, type a "C". The terminal should type:

```
%E222 PLEASE LOGON
/
```

5. The user should then type

```
/LOGON userid#,,C'password'
```

followed by a "CTRL C". The computer will then type various information about the task and return with a slash (/).

PROCEDURE FOR LOGGING ON AN ACOUSTICALLY COUPLED TERMINAL

1. Turn on the acoustic coupler's on/off switch if using a terminal with an external coupler (this switch is usually unmarked).

2. Turn on the power switch on the terminal.

3. Set the PARITY switch on the terminal to EVEN.

4. Set the DUPLEX switch on the coupler and/or terminal to HALF.

5. Set the LOCAL/LINE switch on the coupler and/or terminal to LINE.

Note: On a Teletype Model 33, turn the control knob (below and to the right of the keyboard) counterclockwise instead of steps 2 and 5 above.

6. Set the BAUD switch to the desired position (either 110 or 300 baud -- sometimes denoted as 10 characters/sec. or 30 characters/sec.) on the terminal if so equipped.

7. Dial the computer's telephone number for the baud rate you wish to use (see "pertinent phone numbers"...remember to dial 9 first if using a phone on campus). The phone should ring twice and the computer will answer with a high pitched tone.

8. Upon hearing the tone, immediately place the phone into the cradle. There should be some directive on the coupler indicating at which end the phone cord should be.

9. The CARRIER or SIGNAL light on the coupler will be lit and the terminal will print:

```
%E222 PLEASE LOGON
/
```

10. The user should then type

```
/LOGON userid#,,C'password'
```

followed by a "CTRL C". The computer will then type various information about the task and return with a slash (/).

Provided that the user types a syntactically correct LOGON command containing a valid userid# and password, the system should be ready and waiting.

LOGON PROBLEMS

If the computer doesn't answer, it is probably not operational, and you may obtain a recorded status/schedule report (see "pertinent telephone numbers").

If a busy signal is received, this indicates all available lines are in use. Try again in a few minutes.

If the computer answers, prints nothing, and quickly hangs up, the parity setting is usually incorrect or the phone was incorrectly placed onto the coupler.

If the computer responds normally but the terminal does not print what you type, the DUPLEX setting is incorrect.

The above are some of the more common problems with the LOGON procedure. There are many variations of the above problems which have not been discussed. If your LOGON problem is temporary, that is, you successfully LOGged ON yesterday but have not been able today, feel free to call the UD Office for Computing Activities HELP line (see "pertinent telephone numbers") for help. You could be informing them of a system error. If you have *never* been able to successfully LOGON, please call us, the Engineering and Public Policy Group at or , and we will be happy to discuss your problem.

ISMS-UD FULL TEXT QUERY FACILITY

If the user desires to have programs ISMS-UD and CYCLE present queries in an English text format, two things are necessary. First, a sequential element text file must be created using the Univac file editor EDT. The sequential element text file consists of control information, and the English text for the relational expression and for each element to be considered.

Second, the sequential element text file must be converted into a random access format required by the ISM method. Program MAKEIT performs this function on the sequential element text file and writes the random access format onto a query file.

FORMAT OF THE INPUT SEQUENTIAL ELEMENT TEXT FILE

The format of an example input sequential text file can be seen in its entirety in the Appendix. Certain slash (/) keywords are used to identify the records to be used for the five outputs typed on the terminal for each question when full text queries are used. The five outputs are:

1) introductory clause
2) element a
3) relational clause
4) element b
5) qualifying clause

An example of the above is:

DOES
DEVELOPING SOCIAL INCENTIVES TO LIMIT
HUMAN BIRTHS
HELP
TO ASSURE EACH FAMILY SUFFICIENT LAND FOR
THEIR FOOD NEEDS
IN THE SAHEL REGION OF AFRICA ?

The input sequential text for the above output is:

```
/R1
DOES
/R2
HELP
/R3
IN THE SAHEL REGION OF AFRICA ?
/EL
DEVELOPING SOCIAL INCENTIVES TO LIMIT
HUMAN BIRTHS
/EL
TO ASSURE EACH FAMILY SUFFICIENT LAND FOR
THEIR FOOD NEEDS
// END
```

The slant (/) control keywords define the type of clause for the line(s)* immediately following them. These definitions are listed below.

/R1 = introductory clause

/R2 = relational clause

/R3 = qualifying clause

/EL = element text

// = physical end of sequential element text file

It is important for you to understand that MAKEIT orders the element texts sequentially, starting with one and incrementing by one. That is, element number ten in the list is defined by the text immediately following the tenth /EL card, and so on.

---

*The maximum number of lines is 10. The minimum number of lines is 1.

*The maximum number of characters on one line is 60. The minimum number of characters on one line is 1.

FULL_TEXT_QUERIES_CHECKLIST

To use full text queries, follow this checklist:

1) Using EDT, type in the element list to form a sequential element text file.

2) Run program MAKEIT with the sequential text file created in EDT (during step 1).

3) When typing the "DO" command executing ISMS-UD or CYCLE, be sure to type the name of the sequential text file as the second operand.

4) Answer "Y" to the "FULL TEST QUERIES DESIRED" question that is presented in ISMS-UD or CYCLE.

TEXT_DISPLAYING_HINTS

The following are some of the optional capabilities of the ISMS-UD full text query facility.

CLEARING THE SCREEN OF A CRT

If a CRT (cathode ray tube) terminal is being used for an ISM session, it is possible to have ISMS-UD clear the screen of the display before each query. To do this, all that is necessary is to make the screen clearing character the first character of the introductory clause. The clearing character is usually a CTRL L on most CRTs.

To__clear__the__screen__before__each__query,__type_a_CTRL_L_as_the_first character_of_the_introductory_clause.

SPACING BETWEEN FRAMING CLAUSES

The term "framing clauses" denotes the combination of the introductory clause, relational clause, and qualifying clause. The output using full text queries is always in the form shown on page 10. It is sometimes desirable to display the query with spacing between the framing clauses, e.g.,

DOES

DEVELOPING SOCIAL INCENTIVES TO LIMIT
HUMAN BIRTHS

HELP

TO ASSURE EACH FAMILY SUFFICIENT LAND FOR
THEIR FOOD NEEDS

IN THE SAHEL REGION OF AFRICA ?

in order to increase readablilty. This can be achieved by inserting a line with one blank character 1) after the introductory clause, 2) before and after the relational clause, and 3) before the qualifying clause.

ISMS-UD Version 2.0

July, 1979 University of Dayton

INTERPRETIVE STRUCTURAL MODELING SOFTWARE PACKAGE

The Interpretive Structural Modeling Software Packageconsists of three interactive FORTRAN IV programs which perform operations required to construct and modify Interpretive Structural Models. The following is a description of each program. References to John Warfield's book, *Societal Systems: Planning, Policy, and Complexity*, will be made to help the user locate the theory used.

PROGRAM_MAKEIT

PURPOSE_OF_MAKEIT

Program MAKEIT accepts a sequential element text file, reconstructs it into a random (direct) access format for full text queries, and writes this information onto a query file. The program is conversational in nature to allow the user to display (show) on his or her terminal all elements, or specific pairs of elements, as they might appear during an ISM session.

If you do not know what a sequential element text file is, see "ISMS-UD FULL TEXT QUERY FACILITY".

HOW_TO_USE_MAKEIT

1. To invoke program MAKEIT, use the operating system command /DO MAKEIT. This command has one required operand - the name of the input sequential element text file (i.e., the name used in the EDT @WRITE command). The output query file is automatically named and initialized by prefixing the sequential element text file name with "RND.". The user does not have to worry about this file naming and creation.

   Example.... `/DO MAKEIT,(textfilename)`

   IF...an invalid file name* is typed, the operating system will produce these error messages and will not let you run MAKEIT:

```
% D531 INVALID FILENAME.  COMMAND TERMINATED.
% E015 ERROR IN PRECEEDING CMD - CMDS IGNORED TILL STEP OR LOGOFF.
% E804 ENDPROC RETURNED TO PRIMARY.
```

---

*A valid file name is 1 to 53 characters with no initial numerics and no embedded blanks.

Retype the "/DO" COMMAND.

IF...the syntax of the /DO MAKEIT command is incorrect, the operating system will reject the command and type this error message:

```
% E140 OPERAND SYNTATICAL ERROR: REENTER THE DO COMMAND
```

IF...the /DO MAKEIT command is typed syntactically correct with a valid file name, the operating system will type these messages indicating that MAKEIT is being loaded for use:

```
% P500 LOADING VER# 001 OF ISMS.
 FORTRAN IV PROGRAM MAKEIT  STARTED --- MM/DD/YY
```

2. MAKEIT now begins reading the sequential element text file named in the /DO MAKEIT command.

IF...an unsuccessful read by MAKEIT occurs, the operating system will type this error message:

```
MAKEIT TERMINATED:  TOO MUCH DATA REQUESTED FROM RECORD : P1-CTR
 = NNNNNNNN.
SELECT DEBUG OUTPUT OPTIONS (D,S,A,B,C,HELP)
```

Type a "CTRL C" and the computer will return to the operating system mode enabling the user to correct the problem.

Reasons for an unsuccessful read operation are:

1. the @WRITE'textfilename':1-60: command in EDT was not used.
2. the file name specified in the /DO MAKEIT command was not a sequential element text file.

IF...MAKEIT encounters invalid sequential element text file syntax, these error messages will be printed and MAKEIT will terminate prematurely:

```
***ERROR*** INVALID SLANT KEYWORD ENCOUNTERED
RE-EDIT TEXT FILE TO CORRECT
```

```
***MAKEIT TERMINATED***
```

Re-edit the sequential element text file in EDT and try again.

IF...no problems are encountered, MAKEIT will print this message:

```
PERMFILE HAS BEEN CREATED FOR FULL TEXT QUERIES
SHOW(Y/N) ?
*
```

3. Answer "Y" or "N".

IF...the user types "N", MAKEIT will terminate and return control to the operating system.

IF...the user types "Y", this message will be typed:

```
SHOW ALL ELEMENTS ? (Y/N)
*
```

4. Answer "Y" or "N".

IF...the user types "Y", all of the elements are shown, i.e., printed at the terminal as they might appear during an ISM session. MAKEIT terminates and returns control to the operating system after showing all elements.

IF...the user types "N", this message will be typed:

```
SHOW WHICH ELEMENTS ?
*
```

5. Type the numbers of the two elements that you desire to see. MAKEIT will keep accepting and showing element pairs until zeros are typed. At that time, MAKEIT will terminate and return control to the operating system.

PROGRAM ISMS-UD

PURPOSE OF ISMS-UD

Program ISMS-UD is essentially the hub of the ISMS-UD Version 2.1 software package. ISMS-UD allows the user to embed the ISM utilizing the theory of transitive bordering on a reachability matrix* as well as modify the ISM. Elements that are adjacent on the digraph map, or elements that are on the same level or stage of the hierarchy and are not connected are able to be modified**. ISMS-UD maintains and operates on a reachability matrix which allows a digraph to be obtained after each operation on the matrix. ISMS-UD also allows the user to embed the ISM by using his own queries or the computer's queries.

HOW TO USE ISMS-UD

1. To invoke ISMS-UD, use the operating system command /DO ISMS-UD. This command must include the following two operands: 1) the name of the permanent file which is to contain the ISM, and 2) the name of the sequential element text file for the full text queries option.

   Example:
   ```
   /DO ISMS-UD,(modelfilename,textfilename)
   ```

   IF...full text queries are not going to be used, type "JUNK" for the second operand.

   IF...no model file exists, the operating system will create one and give it the name typed for operand one.

---

*John N. Warfield, *Societal Systems: Planning, Policy and Complexity* (New York: Wiley-Interscience, 1976), p. 237.

**Warfield, p. 356.

IF...a file name is not typed for both operands, the operating system will produce these error messages and will not let you run ISMS-UD:

```
% D531 INVALID FILENAME.  COMMAND TERMINATED.
% E015 ERROR IN PRECEEDING CMD - CMDS IGNORED TILL STEP OR LOGOFF.
% E804 ENDPROC RETURNED TO PRIMARY.
```

IF...an invalid file name is entered, the operating system will respond with one of the following error messages:

```
% E146 ILLEGAL SYMBOLIC PARAMETERS DETECTED WHILE PROCESSING
PROCEDURE FILE; PROCEDURE FILE TERMINATED.
```

or

```
% E144 SYMBOLIC PARAMETER OPERAND ERROR, REENTER THE DO COMMAND.
```

IF...the syntax of the /DO ISMS-UD is incorrect, the operating system will reject the command and type this error message:

```
% E140 OPERAND SYNTATICAL ERROR;  REENTER THE DO COMMAND.
```

After each of these error messages, retype the "/DO" COMMAND.

IF...the /DO ISMS-UD command is correctly typed, the operating system will type these messages indicating that ISMS-UD has been loaded for use:

```
% P500 VER# 2.0 OF ISMS-UD LOADED AT LOCATION 000000.
  FORTRAN IV PROGRAM ISMSUD  STARTED --- MM/DD/YY
```

2. ISMS-UD is now in control and it asks:

```
NEW SYSTEM ? (Y/N)
*
```

A "new system" is an ISM that has never been initialized. Answer "Y" or "N".

IF..."Y" is typed, ISMS-UD will assume a new ISM is being generated and proceed to the next question.

IF..."N" is typed, ISMS-UD will try to read in the ISM from the model file named in operand one of the /DO ISMS-UD command. An unsuccessful attempt to read the ISM will result in one of the following error messages to be typed:

```
ISMSUD  TERMINATED: READ OPERATION ON A NON-EXISTENT ISAM
FILE:  P1-CTR = NNNNNNNN.
SELECT DEBUG OUTPUT OPTIONS (D,S,A,B,C,HELP)

BORD    TERMINATED: ISAM SEQUENTIAL ERROR ODAE: P1- CTR =
NNNNNNNN.
SELECT DEBUG OUTPUT OPTIONS (D,S,A,B,C,HELP)
```

Type a "CTRL C" and the computer will return to the operating system enabling you to retype the /DO ISMS-UD command.

3. Next, ISMS-UD types:

```
TYPE ISMS-UD CMD ? (OR "HELP")
*
```

Type either a two letter ISMS-UD command described below or the keyword "HELP" in order to obtain a list of valid ISMS-UD commands.

ISMS-UD COMMANDS

ISMS-UD recognizes the following two letter keywords as command input:

BO COMMAND

The BO command allows the user to invoke the transitive bordering algorithm to embed an element into the ISM. ISMS-UD will first ask:

```
          FULL TEXT QUERIES DESIRED ? (Y/N)
          *
```

Answer "Y" or "N".

IF...the user types "Y", and the filename for operand two of the /DO ISMS-UD command is: 1) not the name of a sequential element text file, or 2) the name of a sequential element text file that has not undergone the program MAKEIT sequence, the operating system will type this error message:

```
          ISMSUD  TERMINATED: ISAM ERROR OD9A: P1-CTR = NNNNNNNN.
          SELECT DEBUG OUTPUT OPTIONS (D,S,A,B,C,HELP)
```

Type a "CTRL C" and the computer will return to the operating system mode which will enable you to retype the /DO ISMS-UD command.

Next, ISMS-UD asks:

```
          SUBORDINATION RELATION ? (Y/N)
          *
```

Answer "Y" or "N".

IF...you are in doubt, answer "N". A subordination relation is a transitive relation where if Element A is related to Element B, then Element B is not related to Element A. This is an optimization for the transitive bordering algorithm which eliminates the possibility of feedback (cycles) and reduces the number of questions required to embed an element for this bordering session *only*. *Caution*: be sure that you understand the meaning of a subordination relation. In most cases a subordination relation is *not desired*. When in doubt, answer "N".

Note: Once these questions have been answered in an ISM session, they cannot be changed until a TE command is received and a new session started.

The next question is:

```
          TYPE NEXT ELEMENT NUMBER OR 0 FOR BREAK ?
          *
```

Type the number of the first element (if a new system) or the next element (if an old system) to be included in the ISM.

IF...a new system is being modeled, ISMS-UD will form a 1 X 1 reachability matrix and respond with the same question again to get a second element to begin the questioning process.

IF...a non-numeric entry is typed, ISMS-UD will respond with:

```
***ERROR***INPUT NOT NUMERIC--RETRY
*
```

IF...the number typed is already in the system, ISMS-UD will give this error message:

```
***ERROR***  NNNN  ALREADY IN SYSTEM INDEX
```

and ask for another element number.

IF...a 0 (zero) is typed, ISMS-UD will ask for the next command keyword.

IF...a valid element is typed, ISMS-UD will present the queries necessary to embed the element.

When queries are presented, answer with either "Y", "N", or "AB". ISMS-UD will retype the response it received and

IF...a "Y" or "N" is received, ISMS-UD will insert a one or zero, respectively, in the proper location in the reachability matrix and present the next query.

IF..."AB" is received, ISMS-UD will abort the bordering algorithm for the element being processed and ask for the next command keyword.

IF...an invalid response is received, ISMS-UD will issue an error message and present the query again.

After the new element is properly embedded, ISMS-UD will ask for another element number to begin the bordering algorithm again.

BOQ COMMAND

The BOQ command allows the user to invoke the transitive bordering algorithm to embed an element into the ISM by using his or her own queries. (Note: If this is the first time the transitive bordering algorithm is being started, the "full text" and "subordination relation" questions will be asked as in the BO command; however, if they have already been asked, ISMS-UD will assume the answers to be the same as in the BO command.)

Next, ISMS-UD will ask for elements as it did in the BO command. When a valid new element is accepted, ISMS-UD will ask:

```
WHICH ELEMENTS TO BE COMPARED ? (TYPE 0 FOR AUTOMATIC)
*
```

Now type the elements to be used in the query in the order to be compared (i.e. if 2,3 is typed, the query will ask if 2 is related to 3). Remember...the new element <u>must</u> be one of the two elements to be compared.

IF...the relation between the elements typed has already been answered, ISMS-UD will respond with:

```
SORRY, THIS QUESTION HAS BEEN EITHER ASKED ALREADY OR
FILLED BY INFERENCE   EL1 R   EL2 = YES(NO)
```

IF...0 (zero) is typed, ISMS-UD will send a message acknowledging the 0 and will begin to issue queries automatically. The automatic queries option only lasts for the new element. When the next element is entered, ISMS-UD will again ask for elements to be compared.

DI COMMAND

The DI command directs ISMS-UD to extract and print the digraph of the current reachability matrix ISM in a levels format. No other user interaction is required for the execution of this command.

DIS COMMAND

The DIS command directs ISMS-UD to extract and print the digraph of the current reachability matrix ISM in a stages format. No other user interaction is required for the execution of this command.

ADD COMMAND

The ADD command allows the user to add elements to the ISM. This command should be used in conjunction with the AE command to add elements and their relationships. ISMS-UD will type:

```
 TYPE ELEMENTS TO BE ADDED ?
*
```

Type the number of the element to be added. ISMS-UD will keep accepting elements to be added until a zero is typed.

ELIM COMMAND

The ELIM command permits the elimination of elements from the ISM. ISMS-UD types this message:

```
 TYPE ELEMENT NUMBERS TO BE ERASED ?
*
```

Type the number of the element you wish to eliminate from the ISM. ISMS-UD will keep accepting element numbers to be eliminated until a zero is typed.

AE COMMAND

The AE command allows the user to add a relationship (edge) on the ISM. ISMS-UD will type:

```
TYPE <A REACHES TO B>
*
```

Type the two elements to be connected. The edges to be added are accepted until a zero is typed for one or both of the elements.

EE COMMAND

The EE command allows the user to eliminate a relationship (edge) on the ISM. ISMS-UD will type:

```
TYPE <A REACHES TO B> TO BE ERASED
*
```

Type the two elements to be disconnected. The edges to be disconnected are accepted until a zero is typed for one or both of the elements.

IF...either element is not on the minimum edge digraph (i.e. the elements are not connected in any way), ISMS-UD will issue an error message and ask for next command keyword.

IF...the edge entered does not exist (i.e. elements are connected but not directly), ISMS-UD will issue the following error message:

```
***ERROR*** THE EDGE FROM NNN TO MMM DOES NOT

EXIST ON MINIMUM EDGE DIGRAPH
```

IF...the elements typed are in a cycle, ISMS-UD will type:

```
***ERROR*** NNN AND MMM ARE IN A CYCLE
```

"ELIM" NNN AND RE-ENTER USING THE "BO" COMMAND

Eliminate either one of the elements from the system and re-embed using either the BO, BOQ, or ADD and AE command.

PO COMMAND

The PO (Pooling) command permits the user to combine unconnected elements on the same level or stage of a hierarchy into one single element. ISMS-UD will type:

```
TYPE TWO ELEMENTS TO BE POOLED ?
*
```

Type the two elements to be pooled.

IF...the elements to be pooled are not on the same level or stage, ISMS-UD will issue the following error message and ask for two elements to be pooled.

```
***ERROR*** NNN AND MMM ARE NOT ON THE SAME LEVEL

OR STAGE
```

IF...the two elements pass the error checks. ISMS-UD will type

```
NEW INDEX NAME ?
*
```

Type the number that the new element is to take on.

IF...the new index number typed is already in the system, ISMS-UD will inform the user of the error and ask for another new index number.

Elements to be pooled are accepted until two zeros are typed.

EC COMMAND

The EC (Elementary Contraction) command allows the user to combine two adjacent elements on different levels or stages of a hierarchy into one single element. ISMS-UD will type:

```
          TYPE TWO ELEMENTS TO BE CONTRACTED ?
         *
```

Type the two elements to be contracted.

IF...the elements are not adjacent, ISMS-UD will issue the following error message and ask for two elements to be contracted.

```
          ***ERROR*** NNN IS NOT ADJACENT TO MMM
```

IF...it is possible to contract the elements, ISMS-UD will type:

```
          NEW INDEX NAME ?
         *
```

Type the number that the new element is to take on.

IF...the new index number typed is already in the system, ISMS-UD will inform the user of the error and ask for another new index number.

Elements to be contracted are accepted until two zeros are typed.

TE COMMAND

The TE command directs ISMS-UD to terminate and return control to the operating system.

ISMS-UD RESTARTS

Occasionally, problems may arise with a computer system which will render it inoperative. These occurrences are usually infrequent and short. The UD computing facility maintains a 96% up time (i.e., the computer is operational 96% of the scheduled operation time). Even with this excellent record, the restart capabilities of ISMS-UD should be noted. The ISMS-UD software has been designed in such a way that the model permanent file is updated after an operation onto the ISM is completed. The updating operation occurs immediately before this message is typed:

```
TYPE ISMS-UD CMD ? OR ("HELP")
*
```

It can be seen that a small part of the ISM can be lost if the computer crashes (breaks down) while executing any of the embedding or amending commands. Restarting is very easy however. See the procedure below.

How to restart a ISMS-UD session after a time-sharing system failure:

1. Type the /DO ISMS-UD command with the name of the model permanent file and the sequential element text file to be restarted.
2. Answer "N" to the "new system" question. ISMS-UD now has a copy of the ISM.
3. (OPTIONAL) Type "DI" to display the digraph of the restarted ISM.
4. Type the ISMS-UD command you were using when the computer crashed.
5. Continue normally.

PURPOSE_OF_CYCLE

Program CYCLE resolves cycles contained in an Interpretive Structural Model. The user creates and maintains a weighted adjacency matrix that denotes the strength of relationships between elements in the cycle. CYCLE allows the user to obtain listings of the resolution threshold, threshold matrix, and the geodetic paths* .

HOW_TO_USE_CYCLE

1. To invoke CYCLE, use the operating system command /DO CYCLE. This command must include the following two operands: 1) the name of the permanent file which is to contain the weighted matrix, and 2) the name of the sequential element text file for the full text queries option.

   Example... `/DO CYCLE,(weightedmatrixname,textfilename)`

   IF...full text queries are not going to be used, type "JUNK" for the second operand.

   IF...no weighted matrix file exists, the operating system will create one and give it the name typed for operand 1.

   IF...an invalid filename or a filename is not typed for both operands, the operating system will produce these error messages and will not let you run CYCLE:

```
% D531 INVALID FILENAME.  COMMAND TERMINATED.
% E015 ERROR IN PRECEEDING CMD - CMDS IGNORED TILL STEP OR LOGOFF.
% E804 ENDPROC RETURNED TO PRIMARY.
```

---

*Warfield, pp. 328-43

222

Retype the "/DO" COMMAND.

IF...the syntax of the /DO CYCLE command is incorrect, the operating system will reject the command and type this error message:

```
% E140 OPERAND SYNTATICAL ERROR:  REENTER THE DO COMMAND.
```

IF...the /DO CYCLE command is <u>correctly typed</u>, the operating system will type these messages indicating that CYCLE is being loaded for use:

```
% P500 LOADING VER# 001 OF ISMS.
 FORTRAN IV PROGRAM CYCLE   STARTED --- MM/DD/YY
```

2. CYCLE is now in control and it asks:

```
FULL TEXT QUERIES DESIRED ? (Y/N)
*
```

Answer "Y" or "N".

IF...the user types "Y" and the file name for operand two of the /DO CYCLE command is: 1) not the name of a sequential element text file, or 2) the name of a sequential element text file that has not undergone the program MAKEIT sequence, the operating system will type the following error message after attempting to print the first full text query:

```
CYCLE   TERMINATED: ISAM ERROR OD9A: P1-CTR = NNNNNNNN.
SELECT DEBUG OUTPUT OPTIONS (D,S,A,B,C,HELP)
```

Type a "CTRL C" and the computer will return control to the operating system enabling you to retype the /DO CYCLE command.

3. The next question is:

```
NEW SYSTEM ? (Y/N)
*
```

A "new system" is a weighted matrix that has never been initialized. Answer "Y" or "N".

IF..."N" is typed, CYCLE will try to read in the weighted matrix from the file named in operand one of the /DO CYCLE command. An unsuccessful attempt to read in the matrix will result in the following error message to be printed:

```
*EXLST = OPENERR*
```

CYCLE will terminate and return control to the operating system.

IF..."Y" is typed, CYCLE will then type:

```
NUMBER OF ELEMENTS (50 MAX.) ?
*
```

Type the number of elements that are contained in the argument cycle. CYCLE will then ask:

```
REGULAR INDEXING OF ELEMENTS DESIRED ? (Y/N)
*
```

Type "Y" or "N".

IF...the user types "Y", CYCLE assigns sequential numbers to the elements in the cycle. That is, the indexes for the elements are assigned as 1, 2, ... n (where: n is the number of elements).

IF...the user types "N", CYCLE then types:

```
ENTER INDEXES ONE AT A TIME
*
```

The user is now allowed to specify his own index numbers for the elements in the cycle. Type the integers that define the index set one per line.

The advantage of being able to specify index set numbers is apparent when utilizing the full text queries option. For example, suppose elements 3, 7, 59, and 62 of a particular ISM were in a cycle. If full text queries were desired, the user would have to specify his index set as 3, 7, 59, and 62 in order to be able to use the same text file as he did for the embedding session. If "regular indexing" were mistakenly used, the text for elements 1, 2, 3, and 4 would appear whenever full text queries were required and would lead to a great deal of confusion.

4. Next, CYCLE types:

```
TYPE CYCLE COMMAND (OR "HELP")
*
```

Type either a two letter CYCLE command or the keyword "HELP" in order to obtain a list of valid CYCLE commands.

CYCLE_COMMANDS

CYCLE recognizes the following two letter keywords as command input:

AL COMMAND

The AL command allows the user to add elements to the weighted matrix. CYCLE types this message:

```
ELEMENT NUMBER TO BE ADDED ?
*
```

Type the number of the element you wish to add. CYCLE will then ask:

```
BORDER ON THIS ELEMENT ? (Y/N)
*
```

Answer "Y" or "N".

IF...the user types "Y", the weights pertaining to this element can be conveniently assigned by "bordering" down the side and bottom of the weighted matrix. CYCLE will present queries to the user.

IF...the user types "N", the weights are not able to be "bordered" in and must be must be entered via the CH command.

Added element numbers are accepted until a zero is typed.

CH COMMAND

The CH command allows the user to change any weight in the weighted matrix. CYCLE types this message:

```
ENTER A REACHES TO B, WEIGHT (3 NUMBERS)
*
```

Type the element indexes defining the relationship and the new weight. CYCLE will keep accepting changes until three zeros are typed.

The CH command can also be used to initially insert weights into the matrix.

DL COMMAND

The DL command allows the deletion of elements from the weighted matrix. CYCLE types this message:

```
ELEMENT NUMBER TO BE ERASED ?
*
```

Type the number of the element you desire to delete from the weighted matrix. CYCLE will keep accepting element numbers to be deleted until a zero is typed.

FI COMMAND

The FI command facilitates the loading of the weighted adjacency matrix by automatically presenting all the queries necessary to fill the matrix. In addition, if the full text queries option was selected, the FI command will use the text (rather than just the numbers) of the elements during the automatic process. The user should respond to each query by typing an integer (less than 10) which represents the weight of the relationship between the elements.

FI COMMAND RESTARTING. When using the FI command, the user must answer (N**2)-N questions in order to completely fill the weighted matrix (where: "N" is the number of elements in the weighted matrix). A restart/termination feature is available since the number of questions could become quite large. To use the restart feature, simply type a weight of 10 for the element pair you wish to terminate with. CYCLE can now be terminated and resumed at a later time. If the FI command is typed, questioning will resume with the element pair for which the weight of 10 was specified. The restart feature may be used as many times as required.

PR COMMAND

The PR command prints the weighted matrix in its present state on the terminal. No other user interaction is required for the the execution of this command.

RE COMMAND

The RE command is essentially the main command of CYCLE. This command calculates the resolution threshold of the current weighted matrix. The resolution threshold, threshold matrix, and geodetic paths are printed at the terminal. No other user interaction is required for the execution of this command.

TE COMMAND

The TE command directs CYCLE to terminate and return control to the operating system.

APPENDIX_SAMPLE_RUNS

For each of the following sample program runs, the following text file is used for the full text queries option. The text file is presented as the user would type it in using EDT. Comments pertaining to each sample run will immediately follow the listing of the sample run.

EDT_SAMPLE_RUN

```
/EXEC EDT
%  P500 VER# 10B OF EDT LOADED AT LOCATION 000000.


*** EDITOR LOADED (VER 10B), BEGIN TYPE IN:

    1.0000 /R1
    2.0000 DOES
    3.0000 /R2
    4.0000 WEIGH MORE THAN
    5.0000 /R3
    6.0000 IN MOST CASES?
    7.0000 /EL
    8.0000 A BREATH OF AIR  (1)
    9.0000 /EL
   10.0000 AN OLD TENNIS SHOE  (2)
   11.0000 /EL
   12.0000 A FIRE TRUCK  (3)
   13.0000 /EL
   14.0000 A FEATHER  (4)
   15.0000 /EL
   16.0000 AN ELEPHANT  (5)
   17.0000 // END
   18.0000 @WRITE'TEXT.SAMPLE':1-60:
TEXT.SAMPLE IS IN THE CATALOG
OVERWRITE ? (Y,N) Y
```

```
18.0000 @HALT
```

NOTE: If the text file has never been written to the disc, the overwrite message following the "@WRITE" command will not be presented. When the computer returns a slash ("/") after the "@HALT" command, the ISMS-UD programs can be executed.

MAKEIT_SAMPLE_RUN

```
/DO MAKEIT,(TEXT.SAMPLE)
%  P500 VER# 001 OF ISMS LOADED AT LOCATION 000000.
 FORTRAN IV PROGRAM MAKEIT  STARTED --- 04/10/79

 PERMFILE HAS BEEN CREATED FOR FULL TEXT QUERIES
 SHOW(Y/N) ?
*Y


 SHOW ALL ELEMENTS(Y/N) ?
*Y
 DOES
 A BREATH OF AIR  (1)
 WEIGH MORE THAN
 AN OLD TENNIS SHOE  (2)
 IN MOST CASES ?
 DOES
 A FIRE TRUCK  (3)
 WEIGH MORE THAN
 A FEATHER  (4)
 IN MOST CASES ?
 DOES
 AN ELEPHANT  (5)
 WEIGH MORE THAN
 A BREATH OF AIR  (1)
 IN MOST CASES ?
 DOES
 AN OLD TENNIS SHOE  (2)
 WEIGH MORE THAN
 A FIRE TRUCK  (3)
 IN MOST CASES ?
 DOES
 A FEATHER  (4)
 WEIGH MORE THAN
 AN ELEPHANT  (5)
 IN MOST CASES ?
 ** FORTRAN ** CALL EXIT
```

NOTE: If an odd number of elements exists in the text file, as in the preceeding example, MAKEIT will show each element twice so as to display queries until the end of file occurs after a complete query is presented.

ISMS-UD SAMPLE RUN

```
/DO ISMS-UD,(MODEL.SAMPLE,TEXT.SAMPLE)
%  P500 VER# 002 OF ISMS-UD LOADED AT LOCATION 000000.
 FORTRAN IV PROGRAM ISMSUD  STARTED --- 04/10/79

 NEW SYSTEM ? (Y/N)
*Y


 TYPE ISMS-UD CMD ? (OR "HELP")
*HELP



          I S M S  -  U D   C O M M A N D S

      EMBEDING

       BO - TRANSITIVE BORDERING METHOD
      BOQ - TRANSITIVE BORDERING WITH SELECTABLE QUERIES


      DISPLAYING

       DI - DISPLAY MINIMUM EDGE DIGRAPH
            IN A LEVELS FORMAT

      DIS - DISPLAY MINIMUM EDGE DIGRAPH
            IN A STAGES FORMAT


      SUBSTANTIVE AMENDING

      ADD - ADD ELEMENTS

     ELIM - ELIMINATE ELEMENTS

       AE - ADD EDGES (RELATIONSHIPS)
            ON THE MINIMUM EDGE DIGRAPH .
          EE - ERASE EDGES (RELATIONSHIPS)
               ON THE MINIMUM EDGE DIGRAPH


         FORMAT AMENDING

          PO - POOL ELEMENTS

          EC - ELEMENTARY CONTRACTION


         TERMINATION

          TE - TERMINATE ISMS-UD PROGRAM


      ***NOTE***

        HELP - REPRINTS ABOVE LIST


      TYPE ISMS-UD CMD ? (OR "HELP")
     *BO


      FULL TEXT QUERIES DESIRED ? (Y/N)
     *Y
      SUBORDINATION RELATION ? (Y/N)
     *N


      TYPE NEXT ELEMENT NUMBER OR 0 FOR BREAK ?
     *1


      TYPE NEXT ELEMENT NUMBER OR 0 FOR BREAK ?
     *2

       DOES
     A BREATH OF AIR  (1)
     WEIGH MORE THAN
     AN OLD TENNIS SHOE  (2)
     IN MOST CASES ?
     *N N

       DOES
     AN OLD TENNIS SHOE  (2)
     WEIGH MORE THAN
```

```
A BREATH OF AIR  (1)
IN MOST CASES ?
*Y Y


 TYPE NEXT ELEMENT NUMBER OR 0 FOR BREAK ?
*0


 TYPE ISMS-UD CMD ? (OR "HELP")
*DI


           LEVEL NO.    1


                 1


           LEVEL NO.    2


              2 =>    1,


 TYPE ISMS-UD CMD ? (OR "HELP")
*BOQ


 TYPE NEXT ELEMENT NUMBER OR 0 FOR BREAK ?
*3
 WHICH ELEMENTS TO BE COMPARED ? (TYPE 0 FOR AUTOMATIC)
*3,2

  DOES
A FIRE TRUCK  (3)
WEIGH MORE THAN
AN OLD TENNIS SHOE  (2)
IN MOST CASES ?
*Y Y
 WHICH ELEMENTS TO BE COMPARED ? (TYPE 0 FOR AUTOMATIC)
*3,1


 SORRY, THIS QUESTION HAS BEEN EITHER ASKED ALREADY OR FILLED
 BY INFERENCE    3 R    1 = YES
 WHICH ELEMENTS TO BE COMPARED ? (TYPE.0 FOR AUTOMATIC)
```

```
*0
 0 ACKNOWLEDGED, BEGINNING AUTOMATIC QUERIES

  DOES
AN OLD TENNIS SHOE  (2)
WEIGH MORE THAN
A FIRE TRUCK  (3)
IN MOST CASES ?
*N N


 TYPE NEXT ELEMENT NUMBER OR 0 FOR BREAK ?
*0


 TYPE ISMS-UD CMD ? (OR "HELP")
*DI


           LEVEL NO.    1


                  1


           LEVEL NO.    2

               2 =>    1,


           LEVEL NO.    3

               3 =>    2,


 TYPE ISMS-UD CMD ? (OR "HELP")
*ADD


 TYPE ELEMENTS TO BE ADDED ?
*4
*5
*0
```

```
 TYPE ISMS-UD CMD ? (OR "HELP")
*DI


               LEVEL NO.    1


                          1
                          4
                          5


               LEVEL NO.    2


                    2 =>    1,


               LEVEL NO.    3


                    3 =>    2,


 TYPE ISMS-UD CMD ? (OR "HELP")
*AE


 TYPE <A REACHES TO B>
*4,1
*5,3
*3,5
*0,0


 TYPE ISMS-UD CMD ? (OR "HELP")
*DI

   CYCLE ON      3,   5,

               LEVEL NO.    1


                          1
```

```
                LEVEL NO.     2


                    2 =>     1,
                    4 =>     1,


                LEVEL NO.     3


                    3 =>     2,


      TYPE ISMS-UD CMD ? (OR "HELP")
     *EE


      TYPE <A REACHES TO B> TO BE ERASED
     *4,1
     *3,5


      ***ERROR***    3 AND    5 ARE IN A CYCLE.
      "ELIM"    3 AND RE-ENTER USING THE "BO" COMMAND


      TYPE ISMS-UD CMD ? (OR "HELP")
     *DI

        CYCLE ON     3,   5,


                LEVEL NO.     1


                         1
                         4


                LEVEL NO.     2


                    2 =>     1,


                LEVEL NO.     3
```

```
                    3 =>    2,


 TYPE ISMS-UD CMD ? (OR "HELP")
*PO


 TYPE TWO ELEMENTS TO BE POOLED ?
*3,5
 NEW INDEX NUMBER ?
*6


 TYPE TWO ELEMENTS TO BE POOLED ?
*0,0


 TYPE ISMS-UD CMD ? (OR "HELP")
*DI


               LEVEL NO.    1


                      1
                      4


               LEVEL NO.    2


                    2 =>    1,


               LEVEL NO.    3


                    6 =>    2,


 TYPE ISMS-UD CMD ? (OR "HELP")
*EC


 TYPE TWO ELEMENTS TO BE CONTRACTED ?
*2,1
 NEW INDEX NUMBER ?
*7
```

238

```
 TYPE TWO ELEMENTS TO BE CONTRACTED ?
*0,0


 TYPE ISMS-UD CMD ? (OR "HELP")
*DI


               LEVEL NO.    1


                   7
                   4


               LEVEL NO.    2


                 6 =>    7,


 TYPE ISMS-UD CMD ? (OR "HELP")
*ELIM


 TYPE ELEMENT NUMBERS TO BE ERASED ?
*4
*6
*0


 TYPE ISMS-UD CMD ? (OR "HELP")
*DI


               LEVEL NO.    1


                   7


 TYPE ISMS-UD CMD ? (OR "HELP")
*TE
 **FORTRAN ** STOP
```

NOTE: Whenever a 0 (zero) is to be entered to discontinue an ISMS-UD command, a "CTRL C" can be typed instead of the zero to save time;

the computer recognizes this input as a zero. This also holds true for the MAKEIT (when showing which elements) and CYCLE programs.

The above sample run is a *very* simplified example of how each ISMS-UD command affects the ISM. During an actual ISM session, a digraph would not usually be printed until the session is completed. The commands can be used in any logical order.

CYCLE_SAMPLE_RUN

```
/DO CYCLE,(WEIGHT.SAMPLE,TEXT.SAMPLE)
%  P500 VER# 001 OF ISMS LOADED AT LOCATION 000000.
 FORTRAN IV PROGRAM CYCLE   STARTED --- 04/17/79


  FULL TEXT QUERIES DESIRED ? (Y/N)
*Y
  NEW SYSTEM ? (Y/N)
*Y
  NUMBER OF ELEMENTS (50 MAX.) ?
*4
  REGULAR INDEXING OF ELEMENTS DESIRED ? (Y/N)
*Y


  TYPE CYCLE COMMAND (OR "HELP")
*HELP


           ***HELP MESSAGE***

  AL - ADD AN ELEMENT TO SYSTEM
  DL - DELETE AN ELEMENT FROM SYSTEM
  FI - FILL SYSTEM (ASSIGN WEIGHTS)
  CH - CHANGE WEIGHT OF A RELATIONSHIP
  RE - RESOLVE SYSTEM
  PR - PRINT SYSTEM OUT
  TE - TERMINATE SESSION
  HELP - REPRINTS ABOVE LIST


  TYPE CYCLE COMMAND (OR "HELP")
*AL
  ELEMENT NUMBER TO BE ADDED ?
*5
  BORDER ON THIS ELEMENT ? (Y/N)
*Y
 A BREATH OF AIR  (1)
  HAS BEEN RELATED TO
 AN ELEPHANT  (5)
 WEIGHT ?
*9
```

```
 AN OLD TENNIS SHOE  (2)
  HAS BEEN RELATED TO
 AN ELEPHANT  (5)
 WEIGHT ?
*8
 A FIRE TRUCK  (3)
  HAS BEEN RELATED TO
 AN ELEPHANT  (5)
 WEIGHT ?
*7
 A FEATHER  (4)
  HAS BEEN RELATED TO
 AN ELEPHANT  (5)
 WEIGHT ?
*8
 AN ELEPHANT  (5)
  HAS BEEN RELATED TO
 A BREATH OF AIR  (1)
 WEIGHT ?
*4
 AN ELEPHANT  (5)
  HAS BEEN RELATED TO
 AN OLD TENNIS SHOE  (2)
 WEIGHT ?
*5
 AN ELEPHANT  (5)
  HAS BEEN RELATED TO
 A FIRE TRUCK  (3)
 WEIGHT ?
*6
 AN ELEPHANT  (5)
  HAS BEEN RELATED TO
 A FEATHER  (4)
 WEIGHT ?
*2
  ELEMENT NUMBER TO BE ADDED ?
*0


  TYPE CYCLE COMMAND (OR "HELP")
*PR
     1=>    2(0),    3(0),    4(0),    5(9),
     2=>    1(0),    3(0),    4(0),    5(8),
     3=>    1(0),    2(0),    4(0),    5(7),
     4=>    1(0),    2(0),    3(0),    5(8),
     5=>    1(4),    2(5),    3(6),    4(2),
```

242

```
  TYPE CYCLE COMMAND (OR "HELP")
*DL
  ELEMENT NUMBER TO BE ERASED ?
*5
*0


  TYPE CYCLE COMMAND (OR "HELP")
*PR
     1=>    2(0),    3(0),    4(0),
     2=>    1(0),    3(0),    4(0),
     3=>    1(0),    2(0),    4(0),
     4=>    1(0),    2(0),    3(0),


  TYPE CYCLE COMMAND (OR "HELP")
*FI
 A BREATH OF AIR  (1)
  HAS BEEN RELATED TO
 AN OLD TENNIS SHOE  (2)
 WEIGHT ?
*7
 A BREATH OF AIR  (1)
  HAS BEEN RELATED TO
 A FIRE TRUCK  (3)
 WEIGHT ?
*8
 A BREATH OF AIR  (1)
  HAS BEEN RELATED TO
 A FEATHER  (4)
 WEIGHT ?
*9
 AN OLD TENNIS SHOE  (2)
  HAS BEEN RELATED TO
 A BREATH OF AIR  (1)
 WEIGHT ?
*7
 AN OLD TENNIS SHOE  (2)
  HAS BEEN RELATED TO
 A FIRE TRUCK  (3)
 WEIGHT ?
*6
 AN OLD TENNIS SHOE  (2)
  HAS BEEN RELATED TO
 A FEATHER  (4)
 WEIGHT ?
*7
 A FIRE TRUCK  (3)
```

```
  HAS BEEN RELATED TO
 A BREATH OF AIR  (1)
 WEIGHT ?
*0
 A FIRE TRUCK  (3)
  HAS BEEN RELATED TO
 AN OLD TENNIS SHOE  (2)
 WEIGHT ?
*9
 A FIRE TRUCK  (3)
  HAS BEEN RELATED TO
 A FEATHER  (4)
 WEIGHT ?
*7
 A FEATHER  (4)
  HAS BEEN RELATED TO
 A BREATH OF AIR  (1)
 WEIGHT ?
*9
 A FEATHER  (4)
  HAS BEEN RELATED TO
 AN OLD TENNIS SHOE  (2)
 WEIGHT ?
*6
 A FEATHER  (4)
  HAS BEEN RELATED TO
 A FIRE TRUCK  (3)
 WEIGHT ?
*8


  TYPE CYCLE COMMAND (OR "HELP")
*PR
     1=>    2(7),    3(8),    4(9),
     2=>    1(7),    3(6),    4(7),
     3=>    1(0),    2(9),    4(7),
     4=>    1(9),    2(6),    3(8),


  TYPE CYCLE COMMAND (OR "HELP")
*CH
  ENTER A REACHES TO B, WEIGHT (3 NUMBERS)
*3,1,8
*0,0,0


  TYPE CYCLE COMMAND (OR "HELP")
*PR
```

```
          1=>     2(7),     3(8),     4(9),
          2=>     1(7),     3(6),     4(7),
          3=>     1(8),     2(9),     4(7),
          4=>     1(9),     2(6),     3(8),


      TYPE CYCLE COMMAND (OR "HELP")
    *RE


      ***CYCLE RESOLVED***
     THRESHOLD==>    7


                        THRESHOLD MATRIX
          1=>     2(7),     3(8),     4(9),
          2=>     1(7),     4(7),
          3=>     1(8),     2(9),     4(7),
          4=>     1(9),     3(8),

     NUMBER      #LINKS     ELEMENTS     PATH
          1           2        2,    1      2    1    2
          2           2        3,    1      3    1    3
          3           3        3,    2      3    2    1    3
          4           2        4,    1      4    1    4
          5           3        4,    2      4    1    2    4
          6           2        4,    3      4    3    4

      TYPE CYCLE COMMAND (OR "HELP")
    *TE
     ** FORTRAN ** CALL EXIT
```

NOTE: As in the ISMS-UD program, a "CTRL C" can be used instead of zeros when terminating a CYCLE command.

Also as with ISMS-UD, the above example is a very simplified run of the CYCLE program. The user may wish to resolve the matrix several times during the session to obtain the threshold of resolution desired. The commands may also be executed in any logical order as with the ISMS-UD commands.

Table of Contents

Preface . . . 2

Introduction . . . 3

UD time sharing computer system . . . 4
System Hardware . . . 4
System Software . . . 4
Univac File Editor (EDT) . . . 4
Procedures and Important System Concepts . . . 5
  Scheduling the Use of the UD Computer . . . 5
  Automatic LOGOFF Feature . . . 5
  Special Control Keys . . . 5
    End of Transmission Key . . . 5
    Error Correcting Keys . . . 6
  Parity and Duplex Settings for Dial-up Operation . . . 6
  PERTINENT Telephone Numbers . . . 6
How to LOGON . . . 7
  Procedure for LOGging ON a Hardwired Terminal . . . 7
  Procedure for LOGging ON an Acoustically Coupled Terminal . . . 7
  LOGON Problems . . . 9

ISMS-UD Full Text Query Facility . . . 10
Format of the Input Sequential Element Text File . . . 10
Full Text Queries Checklist . . . 12
Text Displaying Hints . . . 12
  Clearing the Screen of a CRT . . . 12
  Spacing between Framing Clauses . . . 12

Interpretive Structural Modeling Software Package . . . 14

Program MAKEIT . . . 15
Purpose of MAKEIT . . . 15
How To Use MAKEIT . . . 15

Program ISMS-UD . . . 18
Purpose of ISMS-UD . . . 18
How To Use ISMS-UD . . . 18
ISMS-UD Commands . . . 20
  BO Command . . . 20
  BOQ Command . . . 23
  DI Command . . . 24
  DIS Command . . . 24

ADD Command 24
ELIM Command 24
AE Command 25
EE Command 25
PO Command 26
EC Command 27
TE Command 27
ISMS-UD Restarts 28

Program CYCLE 29
Purpose of CYCLE 29
How To Use CYCLE 29
CYCLE Commands 32
AL Command 32
CH Command 33
DL Command 33
FI Command 33
FI Command Restarting 34
PR Command 34
RE Command 34
TE Command 34

Appendix A: Sample Runs 35

Makeit Sample Run 36

Isms-ud Sample Run 38

Cycle Sample Run 47

Attachment No. 3

DELTA CHARTS FOR ISMS

Prepared by:

David R. Yingling, Jr.

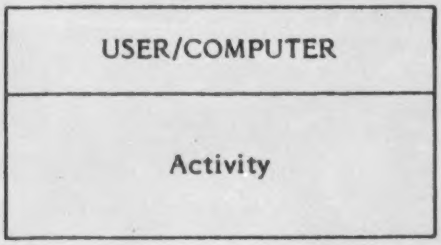
User DELTA Charts

This attachment consists of DELTA charts that describe the user and machine decision interactions that occur while operating ISMS. These DELTA charts are provided for both the programmer and the user in order to facilitate the installation and the use of ISMS. At first glance, you might notice that the DELTA charts look like programming flow charts. However, these charts do not document program flow. Instead, they document the actions of both the user and computer at a typical ISM computer session, therefore, the DELTA charts should be used only for instructional and system overview purposes.

How to Read the DELTA Charts

The charts depict the flow of activities at a typical ISM computer session. They convey a great deal of information in a highly structured format and a relatively small amount of space. This information includes activities, decisions, time flow, logic connections, and who is responsible for each activity or decision.

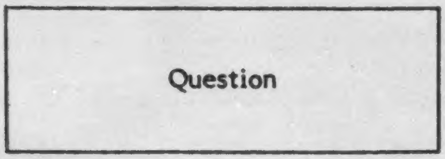
Several symbols appear on the charts; these are explained and illustrated in the next few paragraphs.* Activity boxes are the most common symbol used on the charts. An activity box is divided into two parts; the lower part shows an activity and the top part shows who is responsible for carrying out the activity. In ISM, either the user or the computer is responsible for each activity.



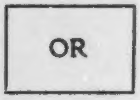
---

*This material is adapted from Warfield (1976, pp. 421-425).

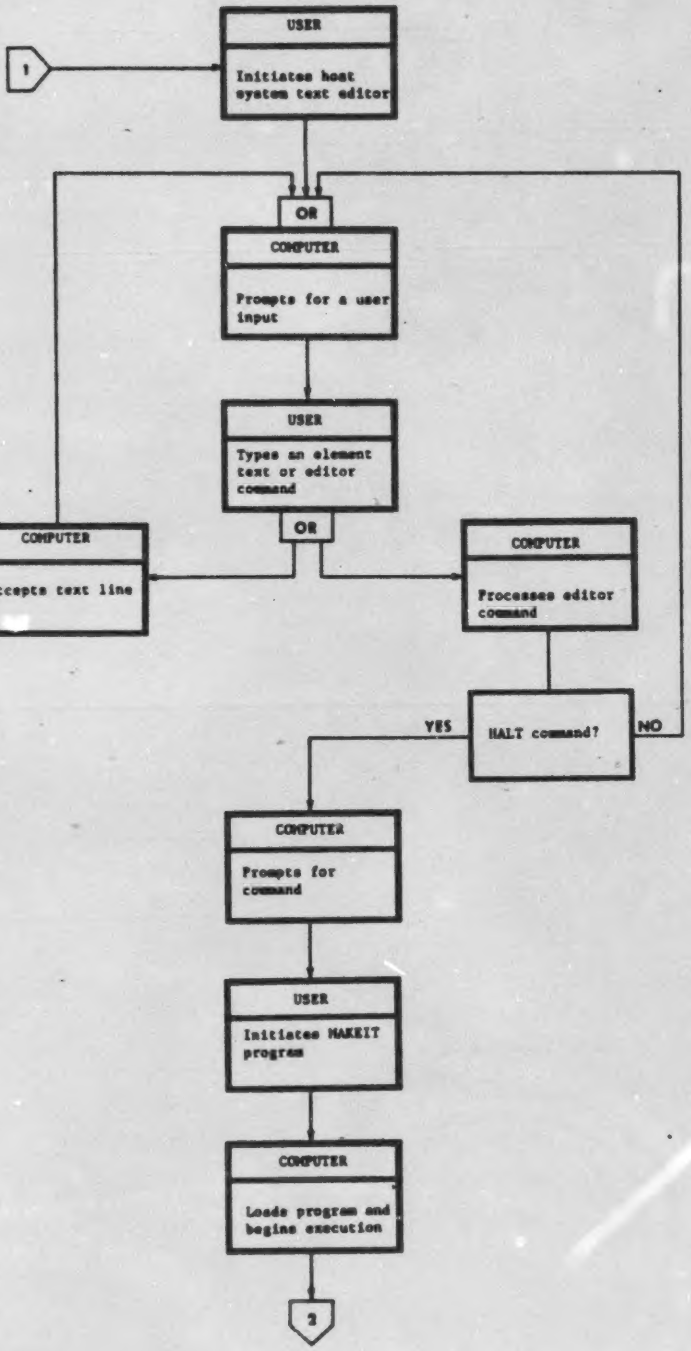
A decision box is similar to an activity box, but it only has one part. By answering the question shown on the decision box, the user chooses among several alternatives paths leading from the decision box to subsequent boxes.

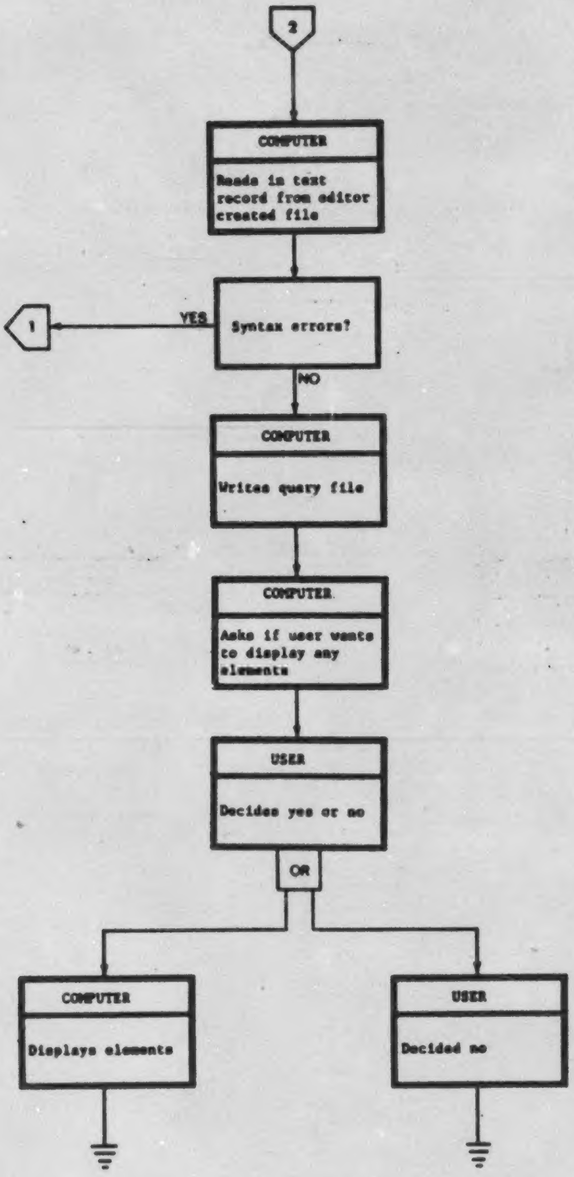


The OR box is interpreted as an "exclusive OR." One and only one of the preceding activities or decisions can occur at a given time. The lines that join the various boxes represent only the flow of time, except at the output of a decision box where lines also represent the various decisions that could be made. In that case, the lines are labeled; usually with either YES or NO.

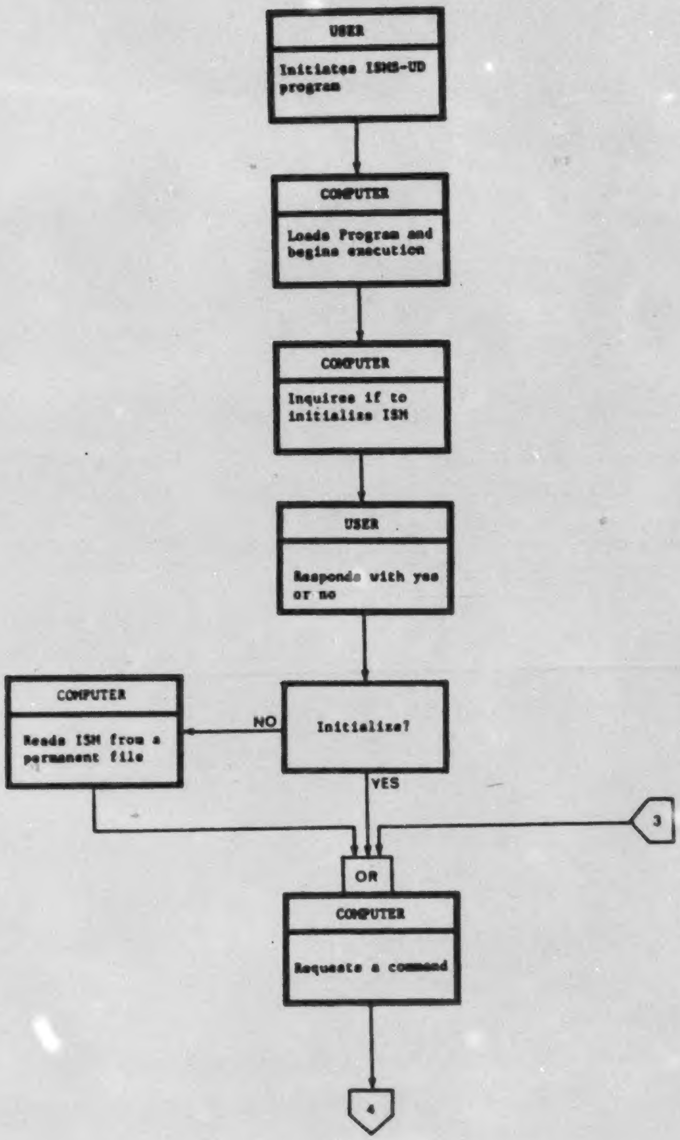


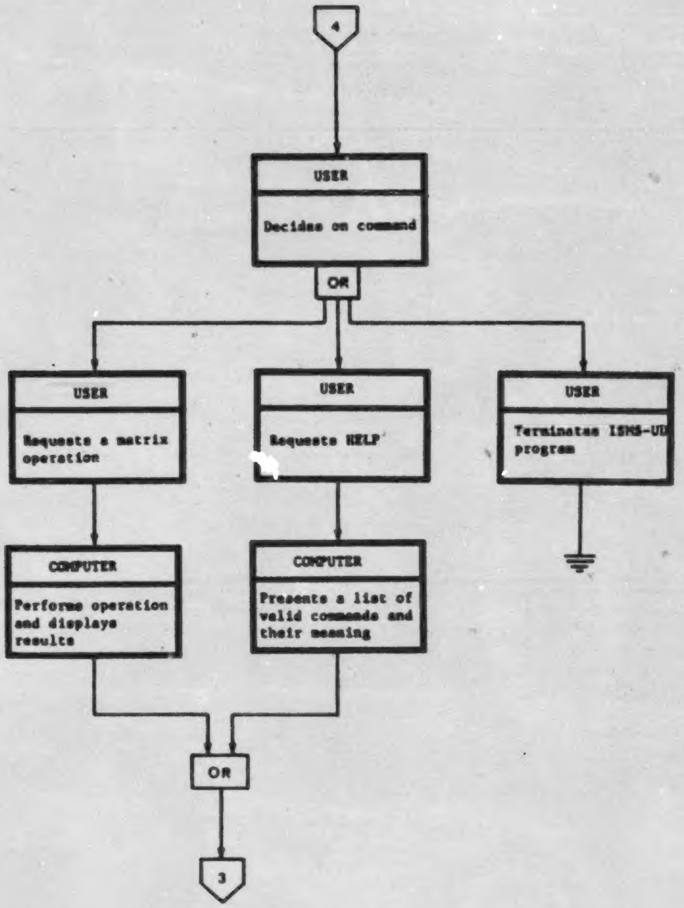
ISMS-US FULL TEXT QUERY FACILITY USE


1
USER
Initiates host system text editor
OR
COMPUTER
Prompts for a user input
USER
Types an element text or editor command
OR
COMPUTER
Accepts text line
COMPUTER
Processes editor command
HALT command?
YES
NO
COMPUTER
Prompts for command
USER
Initiates MAKEIT program
COMPUTER
Loads program and begins execution
2

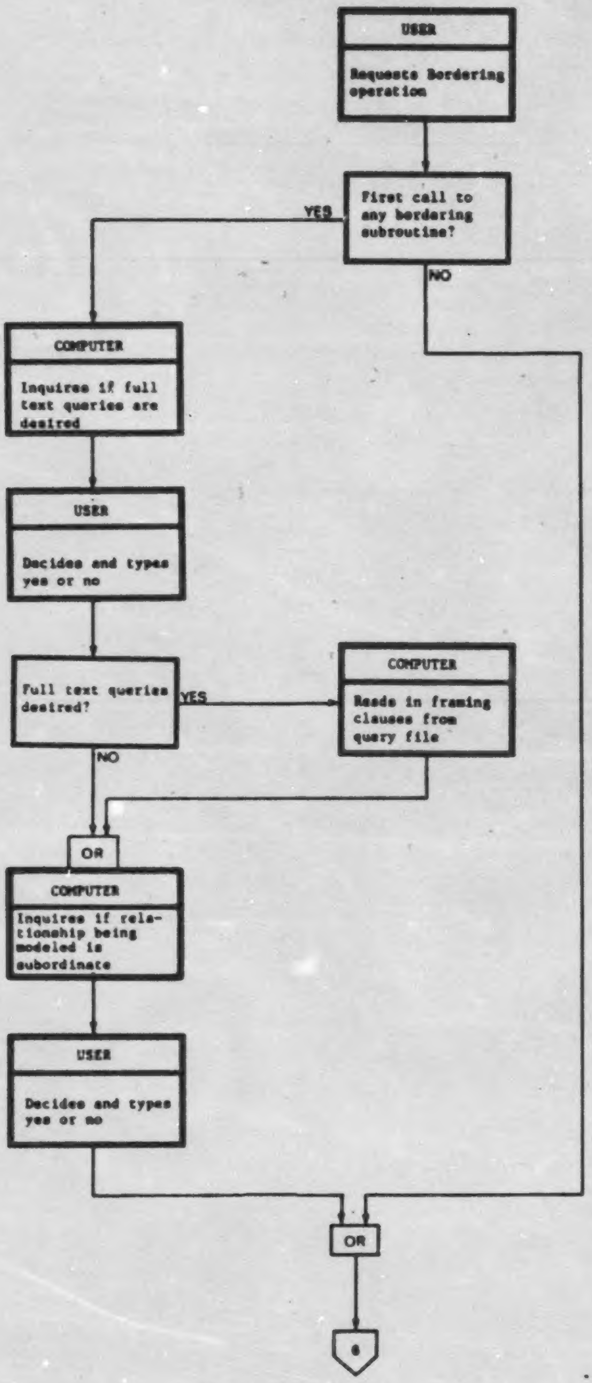

2
COMPUTER
Reads in text record from editor created file
YES
1
Syntax errors?
NO
COMPUTER
Writes query file
COMPUTER
Asks if user wants to display any elements
USER
Decides yes or no
OR
COMPUTER
Displays elements
USER
Decided no

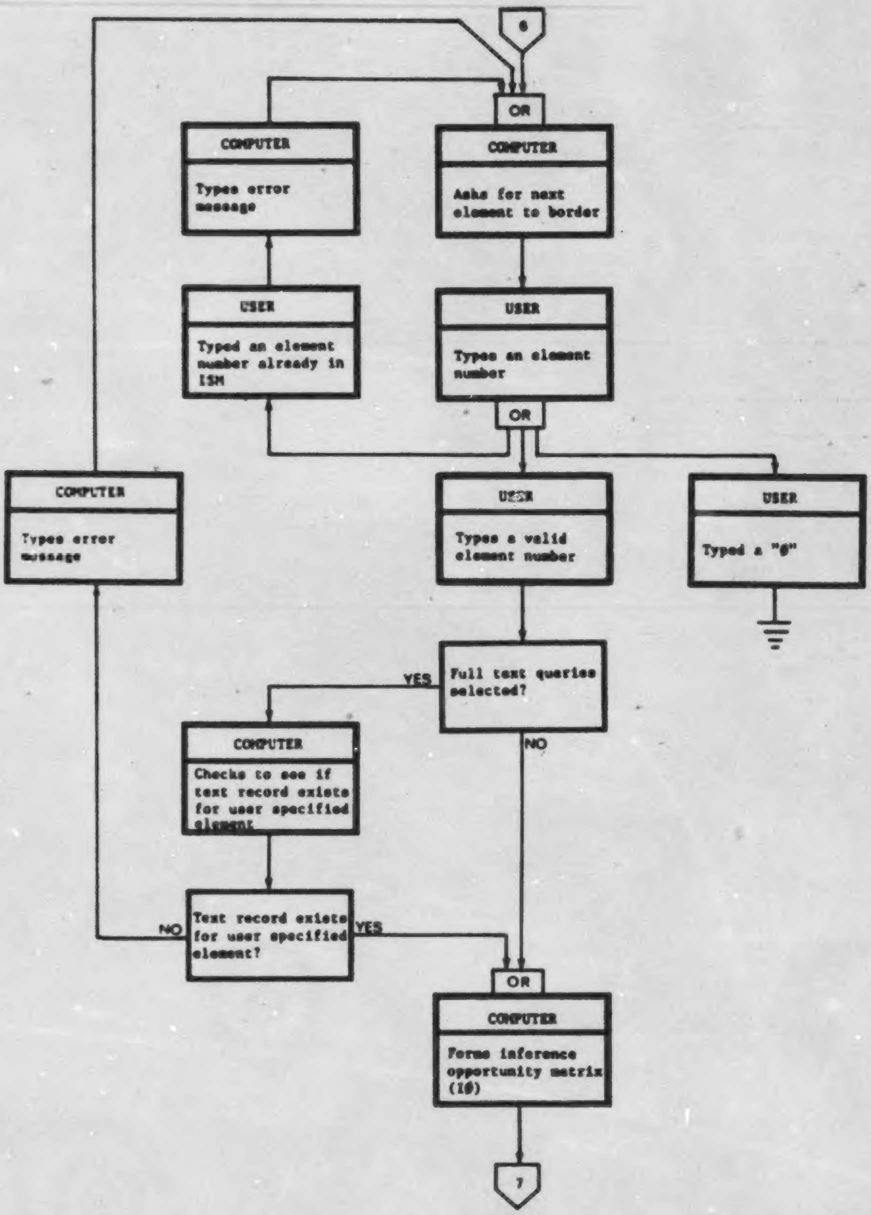
USING PROGRAM ISMS-UD

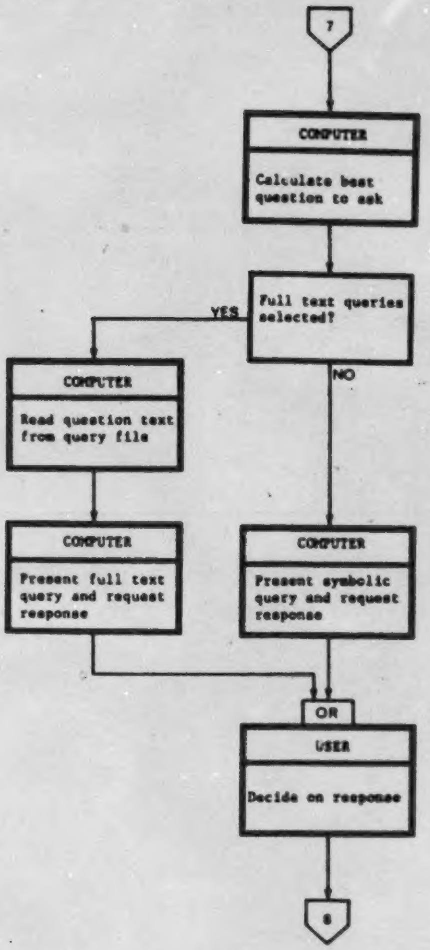

USER
Initiates ISMS-UD program
COMPUTER
Loads Program and begins execution
COMPUTER
Inquires if to initialize ISM
USER
Responds with yes or no
COMPUTER
Reads ISM from a permanent file
NO
Initialize?
YES
3
OR
COMPUTER
Requests a command
4

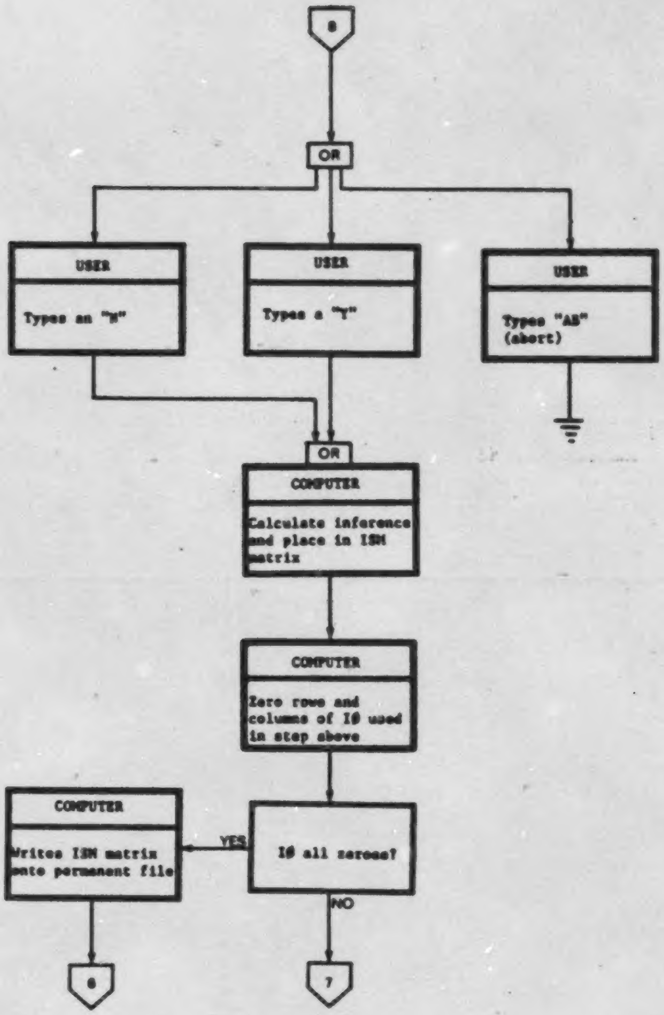

4
USER
Decides on command
OR
USER
Requests a matrix operation
USER
Requests HELP
USER
Terminates ISMS-UD program
COMPUTER
Performs operation and displays results
COMPUTER
Presents a list of valid commands and their meaning
OR
3

BO COMMAND

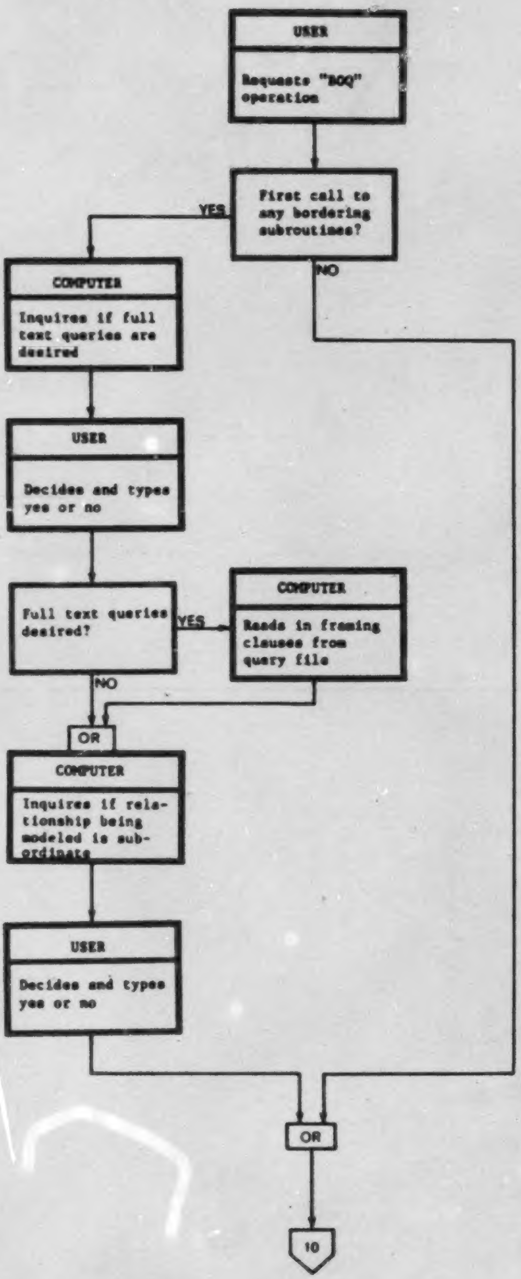

USER
Requests Bordering operation
First call to any bordering subroutine?
YES
NO
COMPUTER
Inquires if full text queries are desired
USER
Decides and types yes or no
Full text queries desired?
YES
NO
COMPUTER
Reads in framing clauses from query file
OR
COMPUTER
Inquires if relationship being modeled is subordinate
USER
Decides and types yes or no
OR

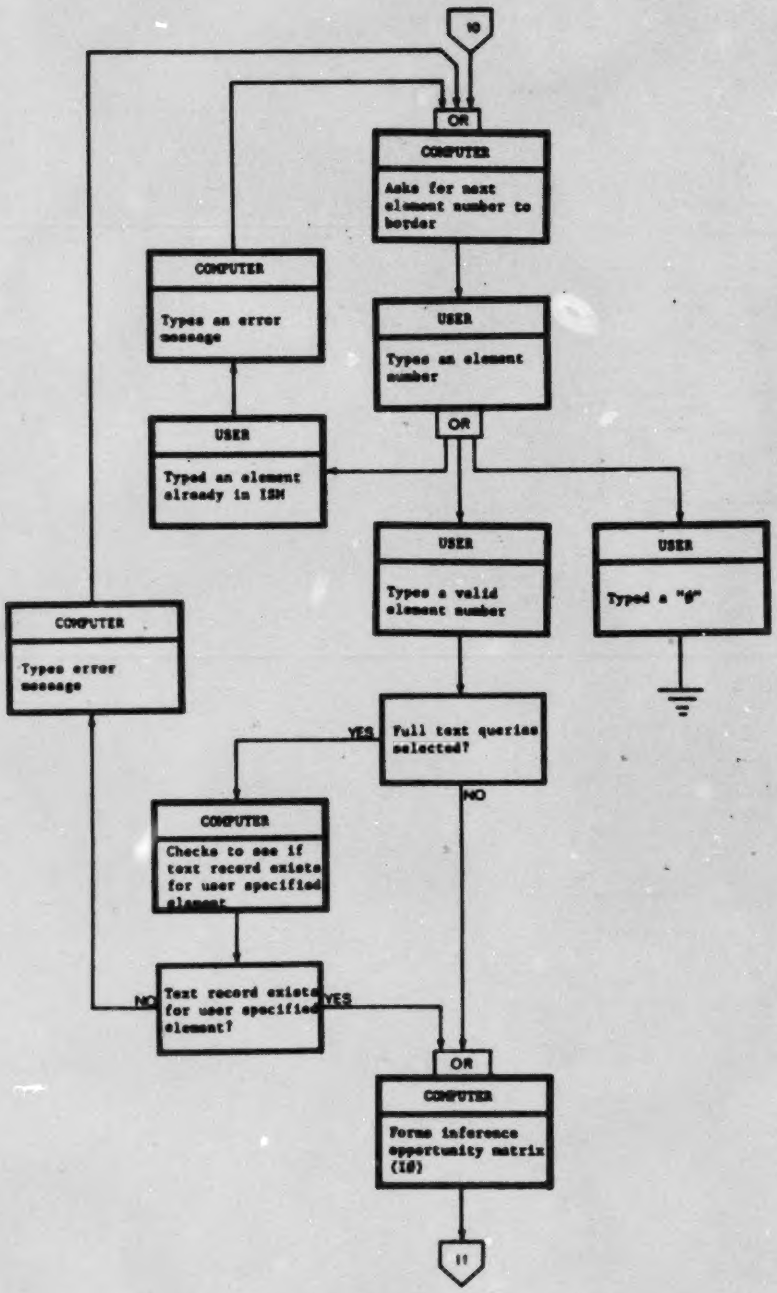

6
OR
COMPUTER
Types error message
COMPUTER
Asks for next element to border
USER
Typed an element number already in ISM
USER
Types an element number
OR
COMPUTER
Types error message
USER
Types a valid element number
USER
Typed a "0"
Full text queries selected?
YES
NO
COMPUTER
Checks to see if text record exists for user specified element
Text record exists for user specified element?
NO
YES
OR
COMPUTER
Forms inference opportunity matrix (I0)
7

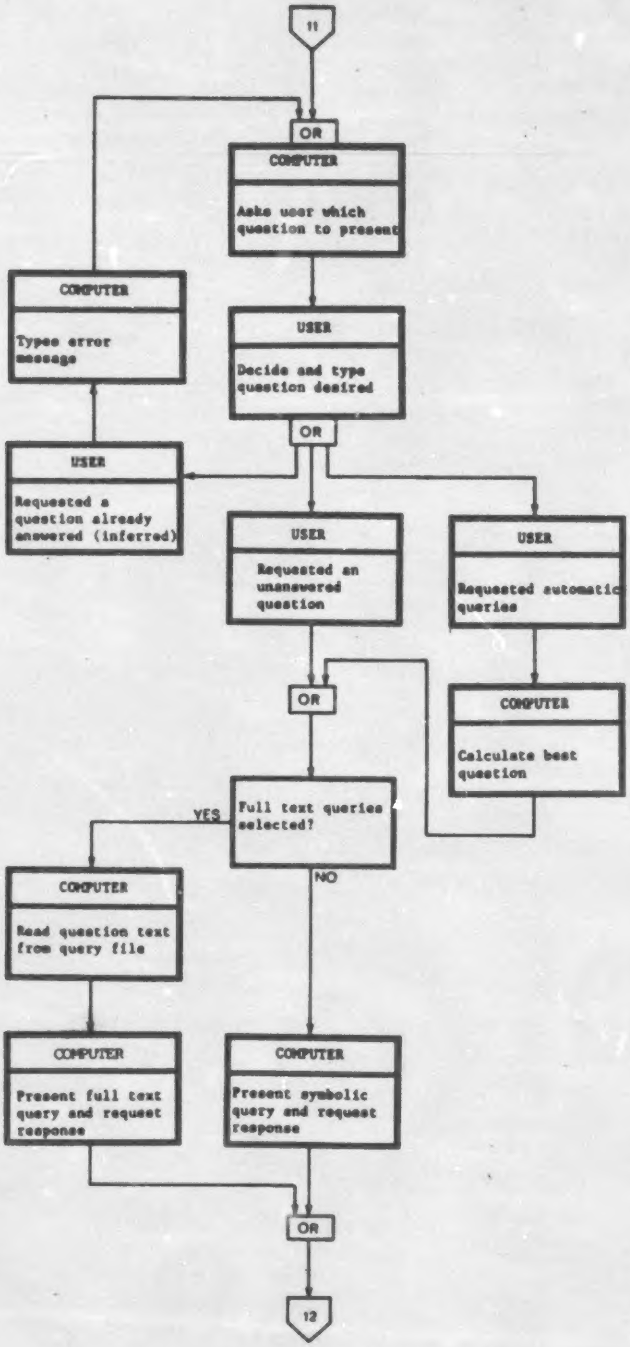

7
COMPUTER
Calculate best question to ask
Full text queries selected?
YES
NO
COMPUTER
Read question text from query file
COMPUTER
Present full text query and request response
COMPUTER
Present symbolic query and request response
OR
USER
Decide on response
8

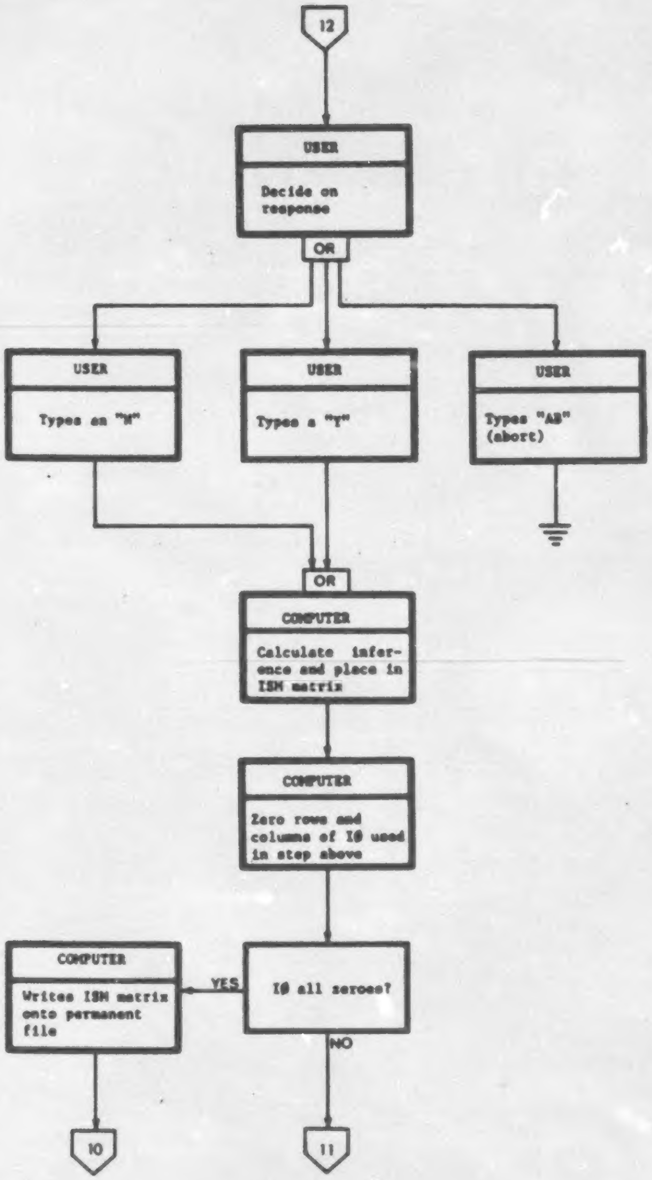

8
OR
USER
Types an "N"
USER
Types a "Y"
USER
Types "AB"
(abort)
OR
COMPUTER
Calculate inference and place in ISM matrix
COMPUTER
Zero rows and columns of I0 used in step above
COMPUTER
Writes ISM matrix onto permanent file
YES
I0 all zeroes?
NO
8
7

BOQ COMMAND

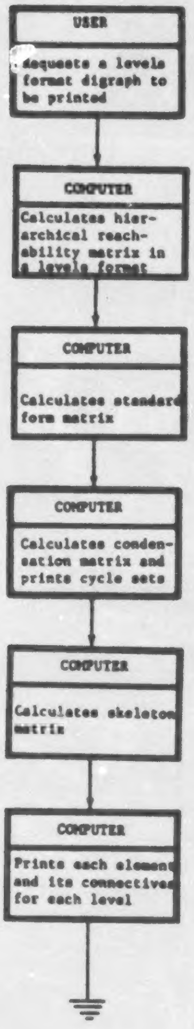

USER
Requests "BOQ" operation
First call to any bordering subroutines?
YES
NO
COMPUTER
Inquires if full text queries are desired
USER
Decides and types yes or no
Full text queries desired?
YES
NO
COMPUTER
Reads in framing clauses from query file
OR
COMPUTER
Inquires if relationship being modeled is subordinate
USER
Decides and types yes or no
OR
10


10
OR
COMPUTER
Asks for next element number to border
USER
Types an element number
OR
COMPUTER
Types an error message
USER
Typed an element already in ISM
USER
Types a valid element number
USER
Typed a "0"
COMPUTER
Types error message
Full text queries selected?
YES
NO
COMPUTER
Checks to see if text record exists for user specified element
Text record exists for user specified element?
NO
YES
OR
COMPUTER
Forms inference opportunity matrix (I0)
11

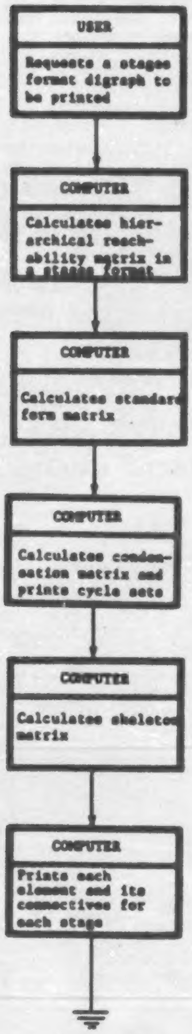

11
OR
COMPUTER
Asks user which question to present
COMPUTER
Types error message
USER
Decide and type question desired
OR
USER
Requested a question already answered (inferred)
USER
Requested an unanswered question
USER
Requested automatic queries
OR
COMPUTER
Calculate best question
Full text queries selected?
YES
NO
COMPUTER
Read question text from query file
COMPUTER
Present full text query and request response
COMPUTER
Present symbolic query and request response
OR
12


12
USER
Decide on response
OR
USER
Types an "N"
USER
Types a "Y"
USER
Types "AB" (abort)
OR
COMPUTER
Calculate inference and place in ISM matrix
COMPUTER
Zero rows and columns of I0 used in step above
I0 all zeroes?
YES
NO
COMPUTER
Writes ISM matrix onto permanent file
10
11

DI COMMAND

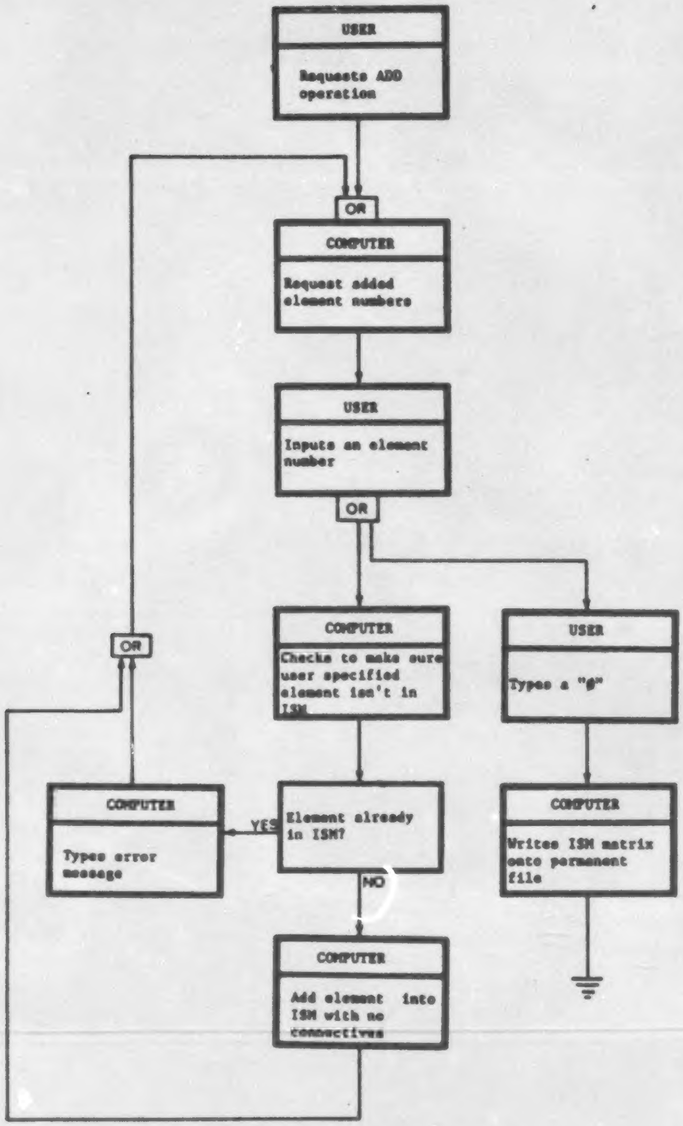

USER
Requests a levels format digraph to be printed
COMPUTER
Calculates hierarchical reachability matrix in a levels format
COMPUTER
Calculates standard form matrix
COMPUTER
Calculates condensation matrix and prints cycle sets
COMPUTER
Calculates skeleton matrix
COMPUTER
Prints each element and its connectives for each level

DIS COMMAND


USER
Requests a stages format digraph to be printed
COMPUTER
Calculates hierarchical reachability matrix in a stages format
COMPUTER
Calculates standard form matrix
COMPUTER
Calculates condensation matrix and prints cycle sets
COMPUTER
Calculates skeleton matrix
COMPUTER
Prints each element and its connectives for each stage

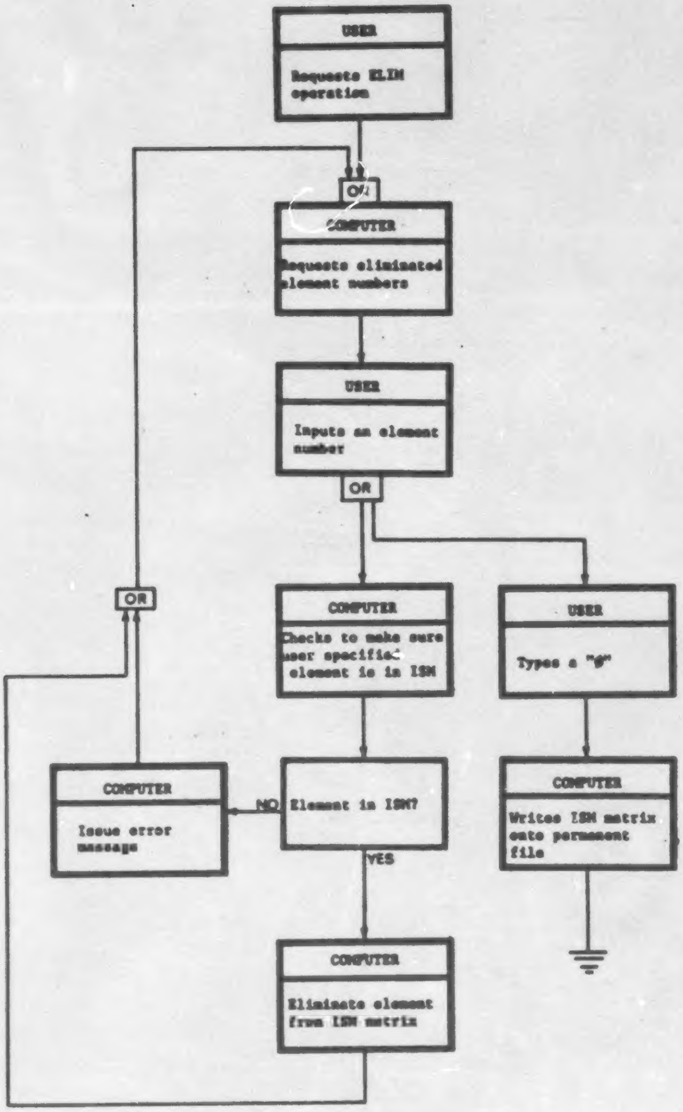
264

ADD COMMAND

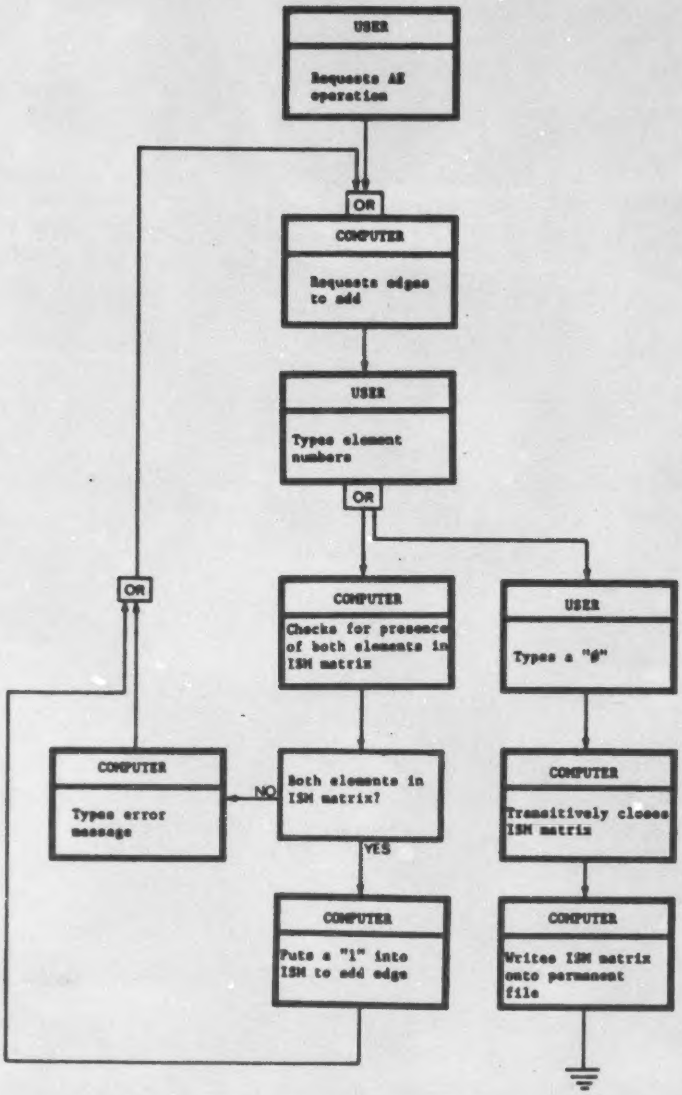

USER
Requests ADD operation
OR
COMPUTER
Request added element numbers
USER
Inputs an element number
OR
COMPUTER
Checks to make sure user specified element isn't in ISM
USER
Types a "Ø"
OR
COMPUTER
Types error message
YES
Element already in ISM?
NO
COMPUTER
Writes ISM matrix onto permanent file
COMPUTER
Add element into ISM with no connectives

265

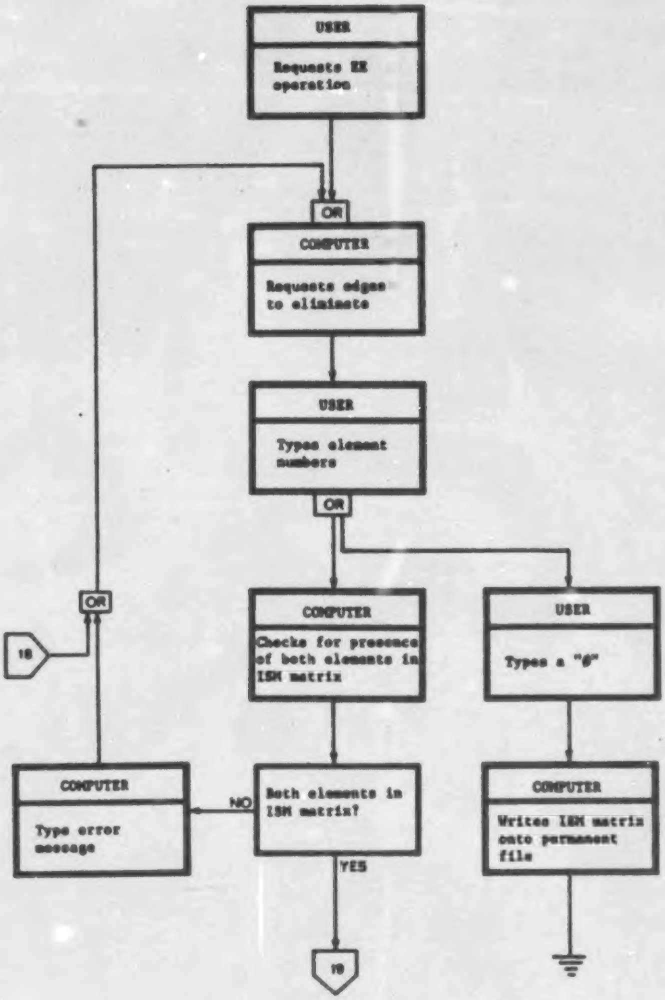
ELIM COMMAND


USER
Requests ELIM operation
OR
COMPUTER
Requests eliminated element numbers
USER
Inputs an element number
OR
COMPUTER
Checks to make sure user specified element is in ISM
USER
Types a "0"
OR
COMPUTER
Issue error message
NO
Element in ISM?
YES
COMPUTER
Writes ISM matrix onto permanent file
COMPUTER
Eliminate element from ISM matrix

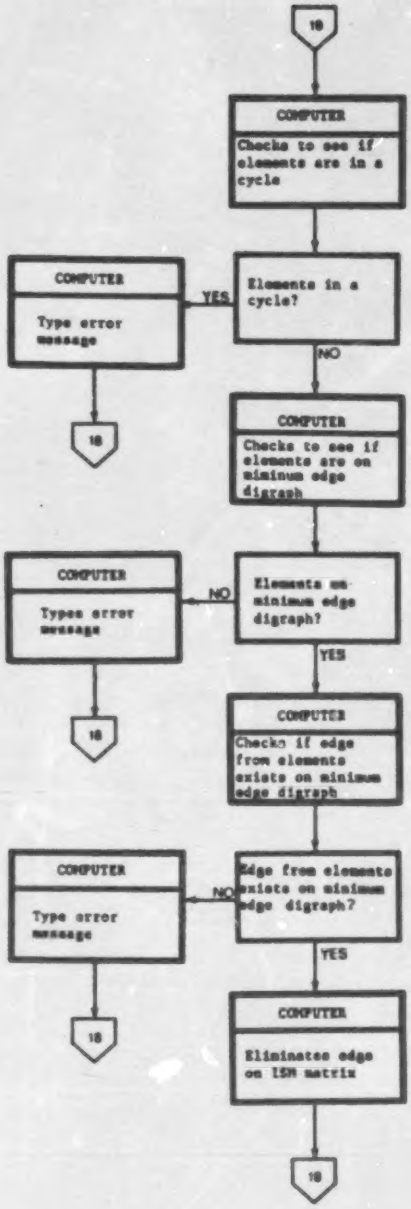
AE COMMAND


USER
Requests AE operation
OR
COMPUTER
Requests edges to add
USER
Types element numbers
OR
COMPUTER
Checks for presence of both elements in ISM matrix
USER
Types a "0"
OR
COMPUTER
Types error message
NO
Both elements in ISM matrix?
YES
COMPUTER
Transitively closes ISM matrix
COMPUTER
Puts a "1" into ISM to add edge
COMPUTER
Writes ISM matrix onto permanent file

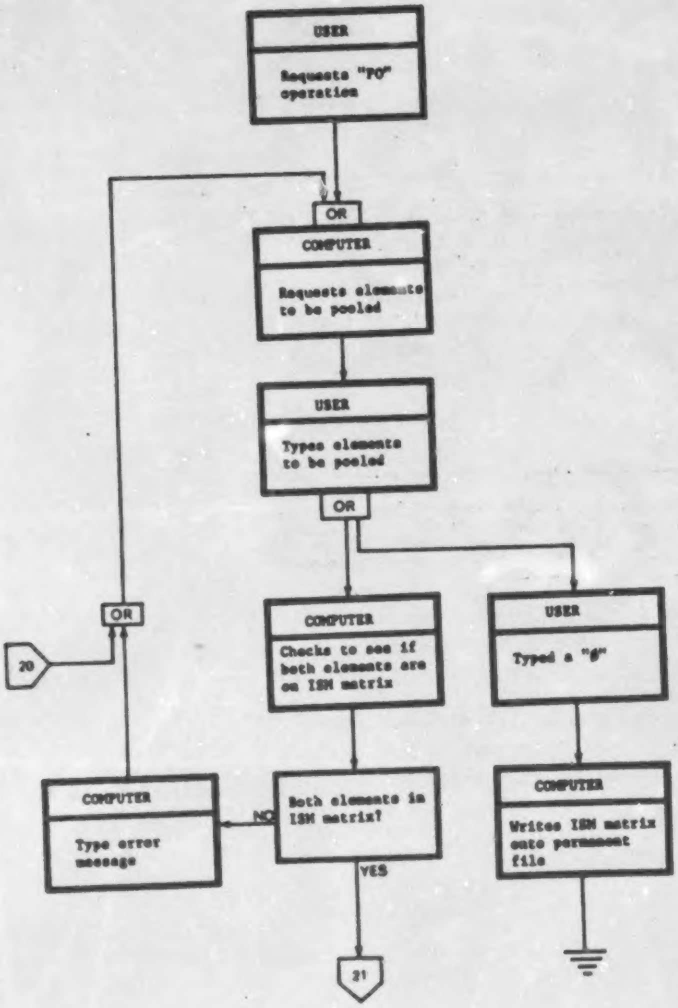
EE COMMAND


USER
Requests EE operation
OR
COMPUTER
Requests edges to eliminate
USER
Types element numbers
OR
COMPUTER
Checks for presence of both elements in ISM matrix
USER
Types a "0"
OR
18
COMPUTER
Type error message
NO
Both elements in ISM matrix?
YES
19
COMPUTER
Writes ISM matrix onto permanent file

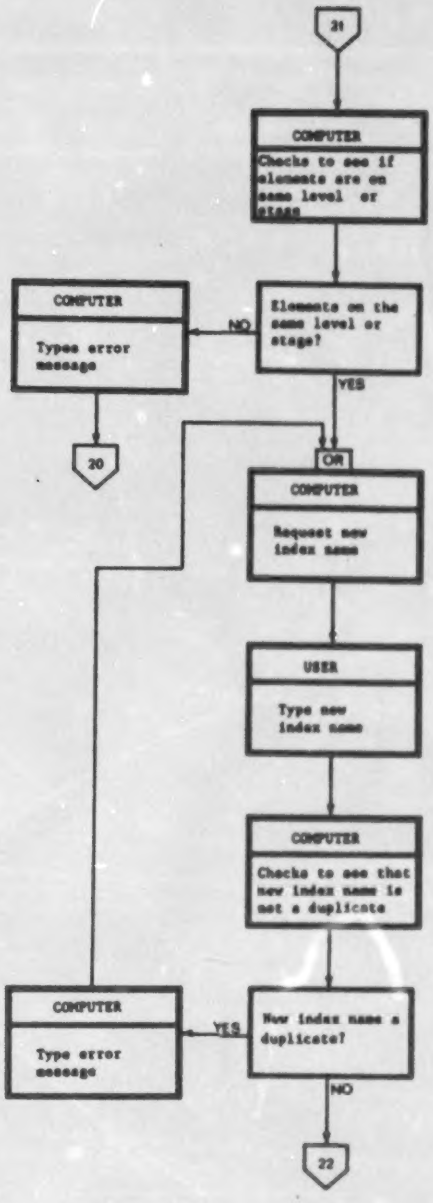
268

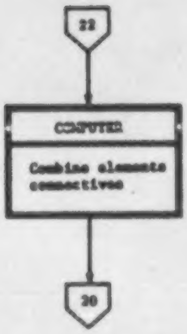

19
COMPUTER
Checks to see if elements are in a cycle
Elements in a cycle?
YES
COMPUTER
Type error message
18
NO
COMPUTER
Checks to see if elements are on minimum edge digraph
Elements on minimum edge digraph?
NO
COMPUTER
Types error message
18
YES
COMPUTER
Checks if edge from elements exists on minimum edge digraph
Edge from elements exists on minimum edge digraph?
NO
COMPUTER
Type error message
18
YES
COMPUTER
Eliminates edge on ISM matrix
18

PO COMMAND

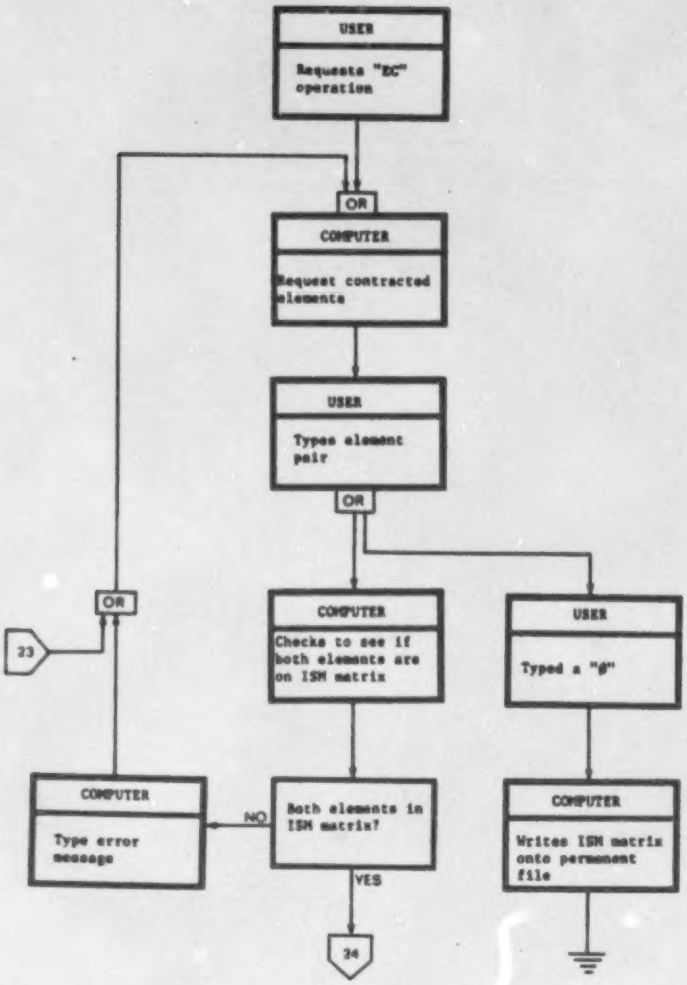

USER
Requests "PO" operation
OR
COMPUTER
Requests elements to be pooled
USER
Types elements to be pooled
OR
COMPUTER
Checks to see if both elements are on ISM matrix
USER
Typed a "0"
OR
20
COMPUTER
Type error message
NO
Both elements in ISM matrix?
YES
COMPUTER
Writes ISM matrix onto permanent file
21

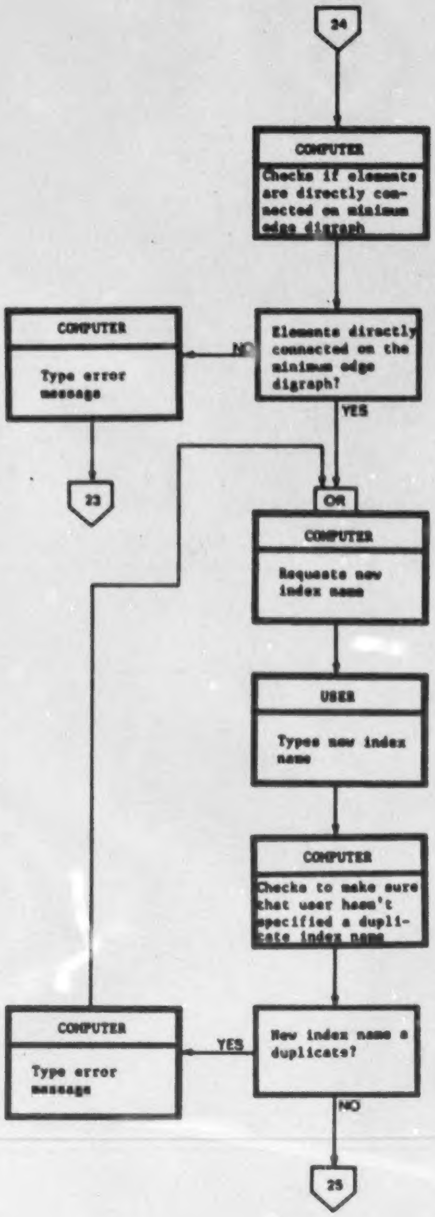
270

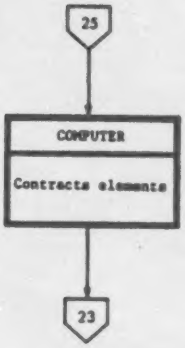

21
COMPUTER
Checks to see if elements are on same level or stage
Elements on the same level or stage?
NO
COMPUTER
Types error message
20
YES
OR
COMPUTER
Request new index name
USER
Type new index name
COMPUTER
Checks to see that new index name is not a duplicate
New index name a duplicate?
YES
COMPUTER
Type error message
NO
22


22
COMPUTER
Combine elements connectives
30

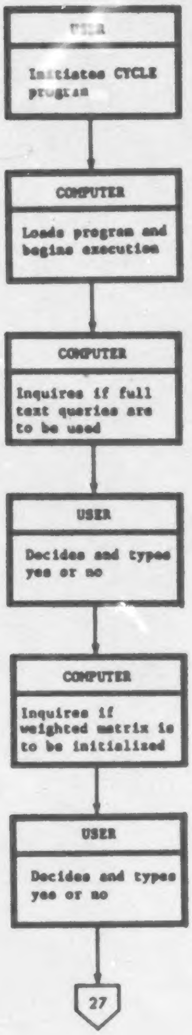
EC COMMAND

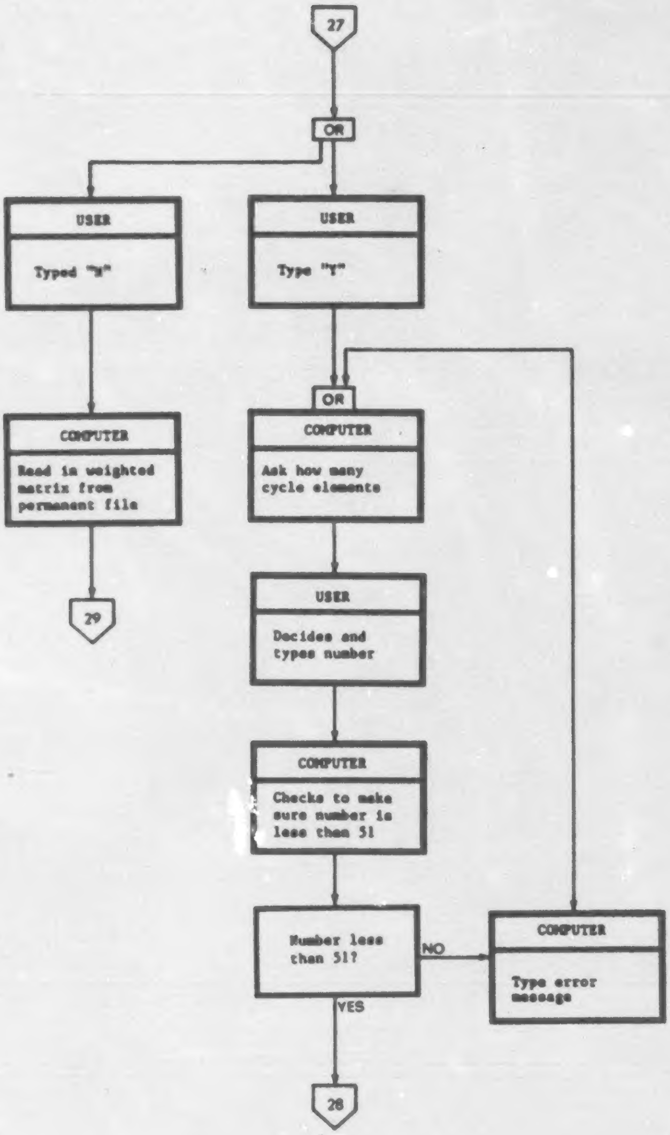

USER
Requests "EC" operation
OR
COMPUTER
Request contracted elements
USER
Types element pair
OR
OR
23
COMPUTER
Checks to see if both elements are on ISM matrix
USER
Typed a "0"
COMPUTER
Type error message
NO
Both elements in ISM matrix?
YES
34
COMPUTER
Writes ISM matrix onto permanent file

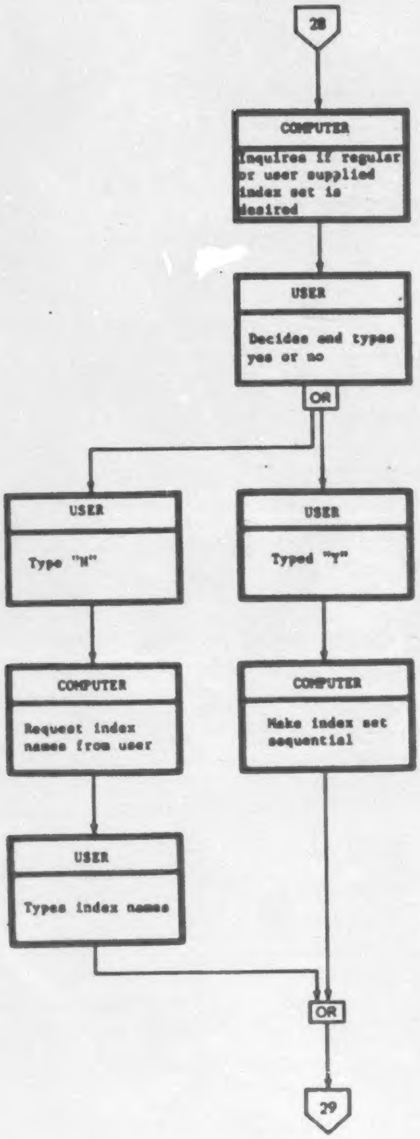

24
COMPUTER
Checks if elements are directly connected on minimum edge digraph
Elements directly connected on the minimum edge digraph?
NO
COMPUTER
Type error message
23
YES
OR
COMPUTER
Requests new index name
USER
Types new index name
COMPUTER
Checks to make sure that user hasn't specified a duplicate index name
New index name a duplicate?
YES
COMPUTER
Type error message
NO
25

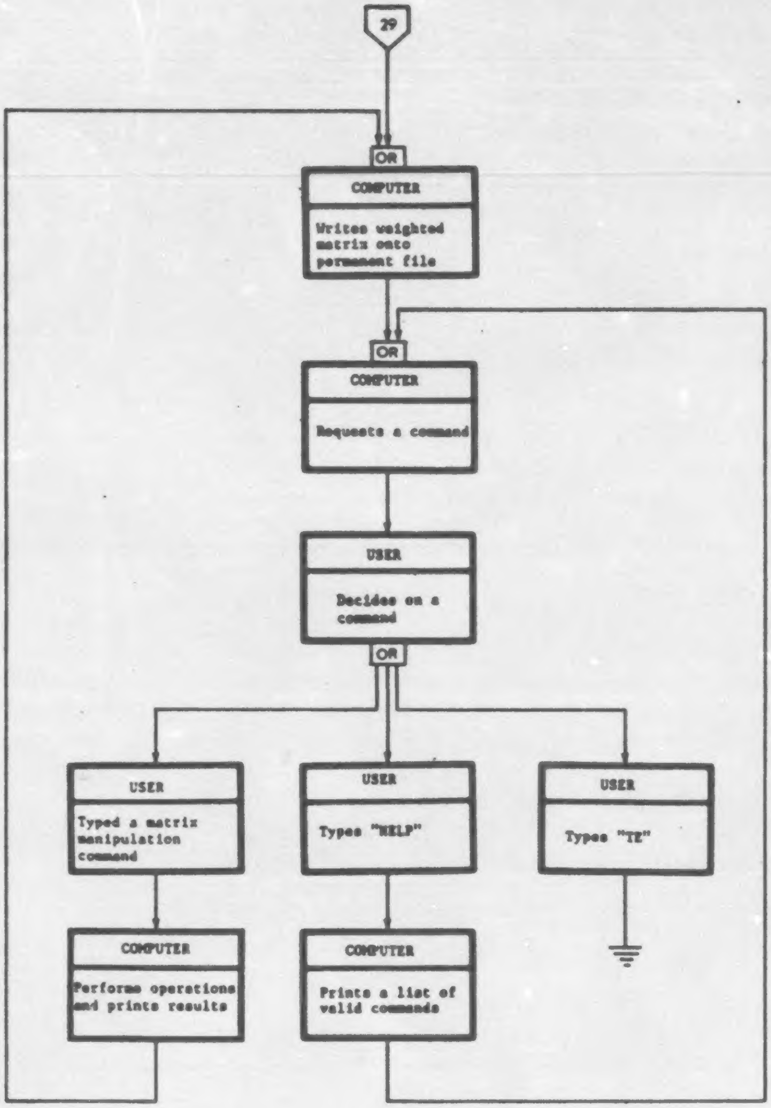

25
COMPUTER
Contracts elements
23

CYCLE PROGRAM


USER
Initiates CYCLE program
COMPUTER
Loads program and begins execution
COMPUTER
Inquires if full text queries are to be used
USER
Decides and types yes or no
COMPUTER
Inquires if weighted matrix is to be initialized
USER
Decides and types yes or no
27


27
OR
USER
Typed "N"
USER
Type "Y"
COMPUTER
Read in weighted matrix from permanent file
29
OR
COMPUTER
Ask how many cycle elements
USER
Decides and types number
COMPUTER
Checks to make sure number is less than 51
Number less than 51?
NO
COMPUTER
Type error message
YES
28


28
COMPUTER
inquires if regular or user supplied index set is desired
USER
Decides and types yes or no
OR
USER
Type "N"
USER
Typed "Y"
COMPUTER
Request index names from user
COMPUTER
Make index set sequential
USER
Types index names
OR
29


29
OR
COMPUTER
Writes weighted matrix onto permanent file
OR
COMPUTER
Requests a command
USER
Decides on a command
OR
USER
Typed a matrix manipulation command
USER
Types "HELP"
USER
Types "TE"
COMPUTER
Performs operations and prints results
COMPUTER
Prints a list of valid commands

UDR-TR-79-79

Appendix to Volume 4:
Conducting Collective Inquiry
COMPUTER IMPLEMENTATION OF INTERPRETIVE STRUCTURAL MODELING
(Part Two)

Written by:

Raymond L. Fitz, S.M.
David R. Yingling
Joanne B. Troha
Karen O. Crim

University of Dayton

September 1979

Attachment No. 4

FORTRAN PROGRAMS FOR ISMS

Prepared by:

David R. Yingling, Jr.

This attachment contains the listing of the FORTRAN IV programs entitled Interpretive Structural Modeling Software (ISMS). Tapes containing the listing of these programs can be obtained for a service fee by writing:

Director
The Office of Computing Activities
University of Dayton
Dayton, Ohio 45469

```
      PROGRAM ISMSUD
C
C     *****************************************************************
C     *                          NOTICE                               *
C     *                                                               *
C     *  ALL RIGHTS RESERVED.  NO PART OF THIS PROGRAM MAY BE SOLD,   *
C     *  REPRODUCED, STORED IN A RETRIEVAL SYSTEM, OR TRANSMITTED     *
C     *  IN ANY FORM OR BY ANY MEANS, ELECTRONIC, MECHANICAL,         *
C     *  PHOTOCOPYING, RECORDING, OR OTHERWISE, WITHOUT THE           *
C     *  PRIOR PERMISSION OF THE                                      *
C     *       UNIVERSITY OF DAYTON RESEARCH INSTITUTE.                *
C     *                                                               *
C     *****************************************************************
C
C                       I  S  M  S    -    U  D
C     INTERPRETIVE STRUCTURAL MODELING SOFTWARE - UNIVERSITY OF DAYTON
C
C     WRITTEN BY:  DAVID R. YINGLING, JR
C                  ENGINEERING AND PUBLIC POLICY GROUP
C                  UNIVERSITY OF DAYTON
C                  DAYTON, OHIO  45469
C
C     VARIABLE NAME                  DESCRIPTION
C
C     N                              THE NUMBER OF ELEMENTS IN THE
C                                    REACHABILITY MATRIX "RM".
C
C     RM                             THE CURRENT REACHABILITY MATRIX
C
C     INDEX                          THE INDEX SET OF "RM".
C
C     NUMS                           A VECTOR USED TO COMMUNICATE WITH
C                                    SUBROUTINE GETNUM
C
C     QTYPE                          LOGICAL VARIABLE WHICH WHEN .FALSE.,
C                                    DENOTES FULL TEXT QUERIES, WHEN
C                                    .TRUE., DENOTES SYMBOLIC QUERIES.
C
C     SUBREL                         LOGICAL VARIABLE WHICH WHEN .TRUE.,
C                                    DENOTES AN OPTIMIZATION OF THE
C                                    TRANSITIVE BORDERING PROCESS IS TO
C                                    BE DONE, WHEN .FALSE., THE
C                                    OPTMIZATION IS NOT ABLE TO BE DONE.
C
C     TTYIN                          FORTRAN READ UNIT NUMBER FOR TELETYPE
C
C     TTYOUT                         FORTRAN WRITE UNIT NUMBER FOR TELETYPE
C
C
C     TXTWDS                         THE NUMBER OF MACHINE WORDS REQUIRED
C                                    TO HOLD ONE OF THE 5 PHRASES OF FULL
C                                    TEXT QUERIES
C
C     SYS                            DIMENSION SIZES OF SYSTEM MATRICES. IT
C                                    IS THE THE MAXIMUM NUMBER OF ELEMENTS
C                                    THAT CAN BE HANDELED BY THE PROGRAM.
C                                    THIS VALUE IS SET BY THE PROGRAMMER
C                                    DEPENDING ON THE AMOUNT OF MEMORY
```

```
C                                   AVAILABLE.
C
C ****COMMON BLOCK  /FTEXT/
C
C      COMMON BLOCK /FTEXT/ IS USED WHEN FULL TEXT QUERIES
C      ARE SELECTED BY THE USER.  IT CONSISTS OF 5 VECTORS
C      DIMENSIONED BY "TXTWDS".  THE MAIN PROGRAM IS RESPONSIBLE
C      FOR READING IN THE "FRAMING" CLAUSES INTO THE BLOCK IN
C      THE POSITIONS R1, R2, AND R3.
C
       IMPLICIT INTEGER*2 (A-Z)
       LOGICAL QTYPE, SUBREL, FIRST, QST
       LOGICAL*1 RM(128,128)
       DIMENSION INDEX(128)
       INTEGER BO,DI,ELIMM,HE,TE,YUP,DIS,AE,EE,ADD,PO,EC,ANSWER,R1,R2,R3
       INTEGER L1,L2,UNUSED,BOQ,N,INDEX
       DATA BO/2HBO/, DI/2HDI/, HE/4HHELP/, TE/2HTE/, YUP/1HY/
       DATA DIS/3HDIS/, AE/2HAE/, EE/2HEE/, ELIMM/4HELIM/, ADD/3HADD/
       DATA PO/2HPO/, EC/2HEC/, BOQ/3HBOQ/
C
       COMMON /FTEXT/ R1(160), L1(160), R2(160), L2(160), R3(160)
C
C      THE BLANK COMMON IS FOR THE U.D. SYSTEM ONLY....
C      THE FORTRAN RUN TIME SUBROUTINES REQUIRE THE ASSOCIATED
C      VARIABLE OF THE DEFINE FILE STATEMENT IN BLANK COMMON
C
       COMMON UNUSED                                                    UDVER2
       SYS    = 128
       NCHAR  = 4
       TXTWDS = 160
       TTYOUT = 2
       TTYIN  = 1
       QTYPE  = .TRUE.
       SUBREL = .FALSE.
       FIRST  = .TRUE.
       QST    = .FALSE.
C
C      NEW SYSTEM ?
C
       WRITE(TTYOUT,101)
       READ (TTYIN, 200)  ANSWER
       IF(ANSWER .EQ. YUP)  GOTO  3
C
C      SYSTEM MARTIX RESIDES ON A PERMFILE, READ IT IN
C
       CALL IO(N,RM,INDEX,10,.TRUE.,SYS)
       GOTO  3
C
C      UPDATE PERMFILE EVERY TIME IN CASE OF TIME
C      SHARING SYSTEM FAILURE
C
     4 CALL IO(N,RM,INDEX,10,.FALSE.,SYS)
C
C      ASK THE USER FOR A ISMSUD COMMAND
C
     3 WRITE(TTYOUT,103)
       READ (TTYIN, 200)  ANSWER
C
```

```
      IF(ANSWER .EQ. BU)     GOTO  5
      IF(ANSWER .EQ. DI)     GOTO  6
      IF(ANSWER .EQ. DIS)    GOTO  9
      IF(ANSWER .EQ. ELIMM)  GOTO  7
      IF(ANSWER .EQ. ME)     GOTO  8
      IF(ANSWER .EQ. AE)     GOTO  10
      IF(ANSWER .EQ. EE)     GOTO  11
      IF(ANSWER .EQ. ADD)    GOTO  12
      IF(ANSWER .EQ. PO)     GOTO  13
      IF(ANSWER .EQ. EC)     GOTO  14
      IF(ANSWER .EQ. BOQ)    GOTO  16
      IF(ANSWER .EQ. TE)     STOP
C
C     INVALID COMMAND
C
      WRITE(TTYOUT,104)  ANSWER
      GOTO  3
C
C     ********************************************************************
C     *           T R A N S I T I V E    B O R D E R I N G               *
C     ********************************************************************
C
    5 IF(.NOT. FIRST)  GOTO  15
      FIRST = .FALSE.
C
C     FULL TEXT QUERIES ?
C
      WRITE(TTYOUT,100)
      READ (TTYIN, 200)  ANSWER
      IF(ANSWER .NE. YUP)  GOTO  1
C
C     FULL TEXT QUERIES DESIRED
C
      DEFINE FILE 8(999,160,U,UNUSED)
C
C     ****NOTE*****
C           THE RECORD SIZE FOR THE ABOVE DEFINE FILE STATEMENT SHOULD
C           BE EQUAL TO "TXTWDS".
C
      FIND(8'2)
      QTYPE = .FALSE.
      READ(8'2)  (R1(I),I=1,TXTWDS)
      READ(8'3)  (R2(I),I=1,TXTWDS)
      READ(8'4)  (R3(I),I=1,TXTWDS)
C
C     SUBORDINATION RELATION ?
C
    1 WRITE(TTYOUT,106)
      READ (TTYIN, 200)  ANSWER
      IF(ANSWER .EQ. YUP)  SUBREL = .TRUE.
C
   15 CALL BORDER(N,RM,INDEX,TTYIN,TTYOUT,QTYPE,SUBREL,QST,TXTWDS,SYS)
      QST = .FALSE.
      GOTO  3
C
C     ********************************************************************
C     *               DISPLAY    DIGRAPH    IN    LEVELS                 *
C     ********************************************************************
```

```
C
    6 CALL DIGLEV(N,RM,INDEX,TTYOUT,SYS)
      GOTO  3
C
C     *******************************************************************
C     *              DISPLAY    DIGRAPH    IN    STAGES                 *
C     *******************************************************************
C
    9 CALL DIGSTG(N,RM,INDEX,TTYOUT,SYS)
C
C     RE-READ N,RM, AND INDEX BECAUSE STAGES ROUTINE
C     HAS USED THEM FOR SCRATCH AREAS
C
      CALL IO(N,RM,INDEX,10,.TRUE.,SYS)
      GOTO  3
C
C     *******************************************************************
C     *                   A D D    A N    E D G E                       *
C     *******************************************************************
C
   10 CALL ADEDGE(N,RM,INDEX,TTYIN,TTYOUT,SYS)
      GOTO  4
C
C     *******************************************************************
C     *                   E R A S E    A N    E D G E                   *
C     *******************************************************************
C
   11 CALL EREDGE(N,RM,INDEX,TTYIN,TTYOUT,SYS)
      GOTO  3
C
C     *******************************************************************
C     *          E L I M I N A T E    A N    E L E M E N T              *
C     *******************************************************************
C
    7 CALL ERASE(N,RM,INDEX,TTYIN,TTYOUT,SYS)
      GOTO  4
C
C     *******************************************************************
C     *                  A D D    A N    E L E M E N T                  *
C     *******************************************************************
C
   12 CALL ADDEL(N,RM,INDEX,TTYIN,TTYOUT,SYS)
      GOTO  4
C
C     *******************************************************************
C     *                  P O O L    E L E M E N T S                     *
C     *******************************************************************
C
   13 CALL POOL(N,RM,INDEX,TTYIN,TTYOUT,SYS)
      GOTO  4
C
C     *******************************************************************
C     *              C O N T R A C T    E L E M E N T S                 *
C     *******************************************************************
C
   14 CALL ELCONT(N,RM,INDEX,TTYIN,TTYOUT,SYS)
      GOTO  4
C
```

```
C     ****************************************************************
C     *          BORDER    WITH    YOUR    OWN    QUERIES             *
C     ****************************************************************
   16 QST = .TRUE.
      GOTO  5
C
C     HELP MESSAGE
C
    8 WRITE(TTYOUT,105)
      GOTO  3
C
C     FORMATS
C
  100 FORMAT(34H-FULL TEXT QUERIES DESIRED ? (Y/N))
  101 FORMAT(19H NEW SYSTEM ? (Y/N))
  103 FORMAT(31H-TYPE ISMS-UD CMD ? (OR "HELP"))
  104 FORMAT(14H-***ERROR***  ,A4,27H <--INVALID ISMS-UD COMMAND/49H IF
     +IN DOUBT TYPE "HELP" FOR LIST OF COMMANDS)
  105 FORMAT(1H-/1H-,5X,34HI S M S  -  U D    C O M M A N D S/1H-,3X,
     +8HEMBEDING/1H0,4X,32HBO - TRANSITIVE BORDERING METHOD/
     +4X,50HBOQ - TRANSITIVE BORDERING WITH SELECTABLE QUERIES/1H-,3X,
     +10HDISPLAYING/1H0,4X,33HDI - DISPLAY MINIMUM EDGE DIGRAPH/10X,
     +18HIN A LEVELS FORMAT/1H0,3X,34HDIS - DISPLAY MINIMUM EDGE DIGRAPH
     +/10X,18HIN A STAGES FORMAT/1H-,3X,20HSUBSTANTIVE AMENDING/1H0,3X,
     +18HADD - ADD ELEMENTS/1H0,2X,25HELIM - ELIMINATE ELEMENTS/1H0,4X,
     +30HAE - ADD EDGES (RELATIONSHIPS)/10X,27HON THE MINIMUM EDGE DIGRA
     +PH/1H0,4X,32HEE - ERASE EDGES (RELATIONSHIPS)/10X,27HON THE MINIMU
     +M EDGE DIGRAPH/1H-,3X,15HFORMAT AMENDING/1H0,4X,18HPO - POOL ELEME
     +NTS/1H0,4X,27HEC - ELEMENTARY CONTRACTION/1H-,3X,11HTERMINATION/
     +1H0,4X,30HTE - TERMINATE ISMS-UD PROGRAM/11H-***NOTE***/1H0,2X,
     +26HHELP - REPRINTS ABOVE LIST)
  106 FORMAT(32H SUBORDINATION RELATION ? (Y/N) )
  200 FORMAT(A4)
      END
      SUBROUTINE BORDER(N,MAT,INDEX,TTYIN,TTYOUT,QTYPE,SUBREL,QST,
     +TXTWDS,SYS)
C
C
C     THIS SUBROUTINE WILL EMBED AN ELEMENT INTO "MAT"
C     UTILIZING THE TRANSITIVE BORDERING ALGORITHM DEVELOPED
C     BY DR. JOHN WARFIELD.
C
C
C     WRITTEN BY:   DAVID R. YINGLING, JR.
C                   ENGINEERING AND PUBLIC POLICY GROUP
C                   UNIVERSITY OF DAYTON
C                   DAYTON, OHIO  45469
C
C     VARIABLE NAME                 DESCRIPTION
C
C     N                             THE NUMBER OF ELEMENTS IN "MAT". UPON
C                                   COMPLETION OF THIS SUBROUTINE,
C                                   "N" = "N" + 1 .
C
C     MAT                           THE CURRENT REACHABILITY MATRIX MODEL
C
C     INDEX                         INDEX SET OF "MAT"
C
C     TTYIN                         FORTRAN READ UNIT NUMBER FOR TELETYPE
C
C     TTYOUT                        FORTRAN WRITE UNIT NUMBER FOR TELETYPE
C
C     QTYPE                         THE QUERY TYPE SWITCH. IF QTYPE =
C                                   .TRUE., SYMBOLIC QUERIES ARE USED.
C                                   IF QTYPE = .FALSE., FULL TEXT QUERIES
C                                   ARE USED.
C
C     SUBREL                        THE SUBORDINATION RELATIONSHIP SWITCH.
C                                   IF THE EMBEDDING RELATIONSHIP IS
C                                   SUBORDINATE, AN OPTMIZATION CAN BE
C                                   MADE THAT REDUCES THE NUMBER OF
C                                   QUESTIONS REQUIRED TO EMBEDD
C                                   ELEMENTS. IF .FALSE., THIS
C                                   OPTMIZATION IS NOT DONE.
C
C     QST                           LOGICAL VARIABLE IF WHEN .TRUE.,
C                                   THE USER IS ABLE TO SELECT HIS
C                                   OWN QUESTIONS. WHEN .FALSE.,
C                                   QUESTIONING IS AUTOMATIC
C
C
C     TXTWDS                        THE NUMBER OF WORDS REQUIRED TO
C                                   HOLD ONE OF THE 5 PHRASES OF
C                                   FULL TEXT QUERIES.
C
C     SYS                           DIMENSION SIZES OF SYSTEM MATRICES
C
C-------
C
C     FLAG                          DENOTES ROWS/COLS OF "PHI" WHICH HAVE
C                                   BEEN ZEROED OUT AS A RESULT OF THE
C                                   SEARCH/ZERO OUT ROUTINES
```

```
C
C     PHI                            THE TRANSITIVE BORDERING INFERENCE
C                                    OPPORTUNITY MATRIX
C
C     Z                              SCRATCH VECTOR USED FOR PHI ZERO OUT
C                                    ROUTINE
C
C     DIMPHI                         THE DIMENSION OF THE CURRENT
C                                    PHI MATRIX
C
C     ZZZ                            LOGICAL SWITCH VARIABLE
C
C
C     ****NOTE****
C     THE DIMENSION SIZES OF "PHI", "Z", AND "FLAG" SHOULD BE
C     SET EQUAL TO 2*"SYS".
C
      IMPLICIT INTEGER*2 (A-Z)
      INTEGER  NOPE,YUP,ABORT,ANSWER,R1,R2,R3,L1,L2,RECORD,UNUSED,N
      INTEGER INDEX, K
      LOGICAL QTYPE, SUBREL, FOUND1, FOUND2, QST, QST1
      LOGICAL*1  FLAG(256), Z(256), PHI(256,256), MAT(SYS,SYS), ZZZ
      DIMENSION INDEX(SYS), NUMS(2)
      DATA NOPE/1HN/, YUP/1HY/, ABORT/2HAB/
C
      COMMON /FTEXT/ R1(160), L1(160), R2(160), L2(160), R3(160)
C
C     THIS BLANK COMMON IS FOR THE U.D. SYSTEM ONLY
C
      COMMON UNUSED                                                     UDVER2
C
C     INITIALIZE EVERYTHING
C
      QST1 = QST
   32 DIMPHI = N * 2
      QST = QST1
C
      DO 1 I=1,DIMPHI
      FLAG(I) = .FALSE.
      Z(I)    = .FALSE.
      DO 1 J=1,DIMPHI
      PHI(I,J) = .FALSE.
    1 CONTINUE
C
C     MAKE SURE SIZE OF ""SYS"" ELEMENTS IS NOT EXCEEDED
C
      IF(N + 1 .LE. SYS)  GOTO  2
      WRITE(TTYOUT,103)  SYS
      GOTO  3
C
C     ASK INTERACTIVE USER FOR NEW ELEMENT NUMBER
C
    2 WRITE(TTYOUT,100)
      CALL GETNUM(NUMS,1,TTYIN,TTYOUT)
      INDEX(N + 1) = NUMS(1)
      IF(INDEX(N + 1) .EQ. 0)  GOTO  3
      IF(N .EQ. 0)  GOTO 35
C
```

```
C     CHECK TO SEE IF USER SPECIFIED AN ELEMENT ALREADY
C     IN THE SYSTEM MATRIX
C
      CALL FINDIT(N,INDEX(N + 1),J,I,J,INDEX,FOUND1,FOUND2,SYS)
      IF(FOUND1)  GOTO  5
   35 IF(QTYPE)  GOTO  6
C
C     FULL TEXT OPTION ON.  MAKE SURE THIS
C     TEXT RECORD IS ON RANDOM TEXT FILE
C
      RECORD = INDEX(N + 1) + 4
      READ(8'RECORD,ERR=36)  (L1(I),I=1,TXTWDS)
      GOTO  6
   36 WRITE(TTYOUT,102)  INDEX(N + 1)
      GOTO  2
C
C     ALL OK, BEGIN XITIVE BORDERING
C     1. FORM PHI MATRIX
C
C
C           A         O
C
C   PHI =
C           -T
C          AA A       A
C
    6 CONTINUE
      IF(N .EQ. 0)  GOTO  33
      SYS2 = 2 * SYS
      CALL TBPHI(N,MAT,PHI,DIMPHI,SUBREL,SYS2,SYS)
C
C     WE NOW HAVE AN INFERENCE OPPORTUNITY MATRIX
C
C     2. DO MIN/MAX PROCEDURE TO FIND THE Z (ZPOINT) WITH
C        THE MOST INFERENCE.
C
    7 CONTINUE
      IF(QST)  GOTO  37
      CALL FINDZ(DIMPHI,PHI,ZPOINT,FLAG,SYS2)
   43 CONTINUE
C
C     SINCE WE NOW HAVE THE Z (ZPOINT) WITH THE MAXIMUM INFERENCE
C     POTENTIAL, ASK THE USER THE RELATION OF THIS ELEMENT WITH THE
C     ADDED ONE.
C
C     IF ZPOINT IS GREATER THEN N, A "Y" QUESTION IS TO BE ASKED
C
C     IF ZPOINT IS LESS THAN OR EQUAL TO N, AN "X" QUESTION
C     IS TO BE ASKED
C
      RELATE = ZPOINT
      IF(ZPOINT .GT. N)  GOTO  8
C
C     ASK THE "X" QUESTION AND CHECK
C
    9 CALL QUEST(INDEX(RELATE),INDEX(N + 1),TTYOUT,QTYPE,TXTWDS)
      READ(TTYIN,200)  ANSWER
      WRITE(TTYOUT,106)  ANSWER
      IF(ANSWER .EQ. ABORT)  GOTO  31
      IF(ANSWER .NE. YUP .AND. ANSWER .NE. NOPE)  GOTO  10
      ZZZ = .TRUE.
      IF(ANSWER .EQ. YUP)  GOTO  11
      GOTO  12
   11 ZZZ = .FALSE.
      GOTO  13
C
C     ERROR MESSAGE
C
   10 WRITE(TTYOUT,104)
      GOTO  9
C
C     ASK THE "Y" QUESTION AND CHECK
C
    8 RELATE = ZPOINT - N
   14 CALL QUEST(INDEX(N + 1),INDEX(RELATE),TTYOUT,QTYPE,TXTWDS)
      READ(TTYIN,200)  ANSWER
      WRITE(TTYOUT,106)  ANSWER
      IF(ANSWER .EQ. ABORT)  GOTO  31
      IF(ANSWER .NE. YUP .AND. ANSWER .NE. NOPE)  GOTO  15
      ZZZ = .FALSE.
      IF(ANSWER .EQ. YUP)  GOTO  16
      GOTO  12
   16 ZZZ = .TRUE.
      GOTO  13
C
C     ERROR MESSAGE
C
   15 WRITE(TTYOUT,104)
      GOTO  8
C
C     COME HERE WHEN THE QUESTION WAS ANSWERED NO.
C
C     IF ZZZ = .TRUE., "X" QUESTION ASKED-SEARCH ROW OF PHI
C
C     IF ZZZ = .FALSE.,"Y" QUESTION ASKED-SEARCH COL OF PHI
C
   12 IF(.NOT. ZZZ)  GOTO  17
      GOTO  18
C
C     COME HERE WHEN THE QUESTION WAS ANSWERED YES
C
C     IF ZZZ = .FALSE.,"X" QUESTION ASKED-SEARCH COL OF PHI
C
C     IF ZZZ = .TRUE., "Y" QUESTION ASKED-SEARCH COL OF PHI
C
   13 IF(.NOT. ZZZ)  GOTO  17
C
C     SEARCH ROW OF PHI FOR INFERENCE TO BE ENTERED INTO "MAT"
C
   18 DO 19 J=1,DIMPHI
      IF(.NOT. PHI(ZPOINT,J))  GOTO  19
      ICTR2 = J
      IF(ICTR2 .GT. N)  ICTR2 = ICTR2 - N
C
      IF(J .LE. N)  GOTO  20
C
```

```
      MAT(N + 1,ICTR2) = .TRUE.
      GOTO 21
C
   20 MAT(ICTR2,N + 1) = .FALSE.
C
   21 FLAG(J) = .TRUE.
      Z(J)    = .TRUE.
   19 CONTINUE
      GOTO  25
C
C     SEARCH COL OF PHI FOR INFERENCE TO BE ENTERED INTO "MAT"
C
   17 DO 22 I=1,DIMPHI
      IF(.NOT. PHI(I,ZPOINT))  GOTO  22
      ICTR2 = I
      IF(ICTR2 .GT. N)  ICTR2 = ICTR2 - N
C
      IF(I .LE. N)  GOTO  23
C
      MAT(N + 1,ICTR2)  = .FALSE.
      GOTO  24
C
   23 MAT(ICTR2,N + 1) = .TRUE.
C
   24 FLAG(I) = .TRUE.
      Z(I)    = .TRUE.
   22 CONTINUE
C
C     PHI MATRIX ZERO OUT ROUTINE
C
   25 DO 26 I=1,DIMPHI
      IF(.NOT. Z(I))  GOTO  26
C
      DO 27 J=1,DIMPHI
      PHI(I,J) = .FALSE.
      PHI(J,I) = .FALSE.
   27 CONTINUE
   26 CONTINUE
C
C     TEST FLAG VECTOR FOR ALL .TRUE.
C     IF SO, ALL ROWS/COLS OF PHI ARE ZERO-ALGORITHM
C     IS COMPLETE-RETURN
C
C     IF NOT ALL .TRUE., ZERO OUT NECESSARY VECTORS
C     AND GO AGAIN.
C
C
      DO 28 I=1,DIMPHI
      IF(.NOT. FLAG(I))  GOTO  29
   28 CONTINUE
C
C     COME HERE WHEN PHI IS NULL
C
   33 N = N + 1
      MAT(N,N) = .TRUE.
C
C     SAVE NEW MATRIX ON PERMFILE IN CASE OF TIMESHARING FALIURE
C
```

```
      CALL IO(N,MAT,INDEX,10,.FALSE.,SYS)
C
C     KEEP BORDERING UNTIL USER TYPES A "0" FOR NEW ELEMT NUMBER
C
      GOTO  32
    3 RETURN
C
C     ERROR MESSAGE
C
    5 WRITE(TTYOUT,101)  INDEX(N + 1)
      GOTO  2
C
C     PHI STILL HAS SOME UNES IN IT
C
   29 DO 30 I=1,DIMPHI
      Z(I)    = .FALSE.
   30 CONTINUE
C
C     GO AGAIN !
C
      GOTO  7
C
C     COME HERE WHEN USER HAS ABORTED BORDERING SEQUENCE
C
   31 WRITE(TTYOUT,105)  INDEX(N + 1)
      GOTO  3
C
C     ********************************
C     ASK YOUR QWN QUESTIONS CODE
C     ********************************
C
   37 WRITE(TTYOUT,107)
      CALL GETNUM(NUMS,2,TTYIN,TTYOUT)
C
C     RESET TO AUTOMATIC QUERIES ?
C
      IF(NUMS(1) .LE. 0 .OR. NUMS(2) .LE. 0)  GOTO  41
C
C     ONE OF NUMBERS TYPES MUST BE EQUAL TO INDEX(N + 1)
C
      IF(NUMS(1) .NE. INDEX(N + 1) .AND. NUMS(2) .NE. INDEX(N + 1))
     +GOTO  38
C
C     NEXT CHECK IS JUST IN CASE USER TRIES SOME FUNNY STUFF
C
      IF(NUMS(1) .EQ. INDEX(N + 1) .AND. NUMS(2) .EQ. INDEX(N + 1))
     +GOTO  37
C
C     WE KNOW THAT ONE OF THE NUMBERS READ IN IS THE NEW ELEMENT
C     BEING ADDED.  FIND OUT WHICH ONE IT IS AND SEE IF OTHER
C     NUMBER TYPED IS IN SYSTEM INDEX
C
      IF(NUMS(1) .EQ. INDEX(N + 1))  GOTO  39
C
C     IF WE MAKE IT HERE, SECOND NUMBER TYPED IS NEW ELEMENT.
C     SEE IF FIRST ONE IS IN SYSTEM INDEX
C
      K = 1
   42 CALL FINDIT(N,NUMS(K),J,I,J,INDEX,FOUND1,FOUND2,SYS)
      IF(FOUND1)  GOTO  40
C
C     ERROR MESSAGE, ONE OF NUMBERS TYPED IS NOT IN SYSTEM INDEX
C
      WRITE(TTYOUT,108)  NUMS(K)
      GOTO  37
C
C     RESET BACK TO AUTOMATIC QUERIES
C
   41 QST = .FALSE.
      WRITE(TTYOUT,110)
      GOTO  7
C
C     ERROR MESSAGE, ONE OR THE OTHER OF THE NUMBERS TYPED MUST BE THE
C                    NEW ELEMENT BEING INTRODUCED INTO SYSTEM
C
   38 WRITE(TTYOUT,109)  INDEX(N + 1)
      GOTO  37
C
C     IF WE MAKE IT HERE, FIRST NUMBER IS NEW ELEMENT.
C     SEE IF SECOND TYPED IS IN SYSTEM INDEX
C
   39 K = 2
      GOTO  42
C
C     NOW CHECK TO MAKE SURE THAT USER HASN'T' EITHER ALREADY
C     ANSWERED THIS QUESTION OR INFERED THE ANSWER
C
   40 ZPOINT = I
      IF(K .EQ. 2)  ZPOINT = ZPOINT + N
      IF(.NOT. FLAG(ZPOINT))  GOTO  43
C
      K = N + 1
      CALL FINDIT(K,NUMS(1),NUMS(2),I,J,INDEX,FOUND1,FOUND2,SYS)
      IF(MAT(I,J))  GOTO  44
      WRITE(TTYOUT,111)  NUMS(1), NUMS(2)
      GOTO  37
   44 WRITE(TTYOUT,112)  NUMS(1), NUMS(2)
      GOTO  37
C
C     FORMATS
C
  100 FORMAT(42H-TYPE NEXT ELEMENT NUMBER OR 0 FOR BREAK ?)
  101 FORMAT(12H-***ERROR***,I5,24H ALREADY IN SYSTEM INDEX)
  102 FORMAT(29H-***ERROR*** TEXT FOR ELEMENT,I5,17H NOT IN TEXT FILE)
  103 FORMAT(62H-***NOTE*** NUMBER OF ELEMENTS HAS REACHED COMPUTER'S LI
     +MIT OF,I5/24H "BO" COMMAND TERMINATED)
  104 FORMAT(46H-***ERROR*** INVALID RESPONSE TO LAST QUESTION)
  105 FORMAT(42H-***NOTE*** BORDERING SEQUENCE FOR ELEMENT,I5,17H HAS BE
     +EN ABORTED)
  106 FORMAT(1H+,2X,A2)
  107 FORMAT(55H WHICH ELEMENTS TO BE COMPARED ? (TYPE 0 FOR AUTOMATIC))
  108 FORMAT(12H-***ERROR***,I5,20H NOT IN SYSTEM INDEX)
  109 FORMAT(12H-***ERROR***,I5,28H MUST BE ONE OF THE ELEMENTS)
  110 FORMAT(43H 0 ACKNOWLEDGED, BEGINING AUTOMATIC QUERIES)
  111 FORMAT(64H-SORRY, THIS QUESTION HAS BEEN EITHER ASKED ALREADY OR F
     +ILLED BY/10H INFERENCE,I5,2H R,I5,5H = NO)
  112 FORMAT(64H-SORRY, THIS QUESTION HAS BEEN EITHER ASKED ALREADY OR F
     +ILLED BY/10H INFERENCE,I5,2H R,I5,6H = YES)
  200 FORMAT(A2)
      END
      SUBROUTINE FINDZ(DIMPHI,PHI,ZPOINT,FLAG,SYS2)
C
C     THIS SUBROUTINE RETURNS THE Z (ZPOINT) WITH THE MAXIMUM
C     INFERENCE POTENTIAL GIVEN A PHI MATRIX
C
C     WRITTEN BY: DAVID R. YINGLING, JR.
C                 ENGINEERING AND PUBLIC POLICY GROUP
C                 UNIVERSITY OF DAYTON
C                 DAYTON, OHIO 45469
C
C     VARIABLE NAME                     DESCRIPTION
C
C     DIMPHI                            THE CURRENT DIMENSION OF "PHI".
C
C     PHI                               THE TRANSITIVE BORDERING INFERENCE
C                                       OPPORTUNITY MATRIX.
C
C     ZPOINT                            THE ROW/COL OF "PHI" WHICH
C                                       HAS MAXIMUM INFERENCE POTENTIAL
C
C     FLAG                              VECTOR DENOTING ROWS/COLS OF "PHI"
C                                       WHICH HAVE BEEN SET TO ALL ZEROS
C
C     SYS2                              DIMENSION SIZES OF SYSTEM MATRICES
C
C-------
C
C     MIN                               SCRATCH VECTOR CONTAINING THE
C                                       MIN2(V1,V2) SET
C
C     V1                                VECTOR CONTAINING THE NUMBER OF ONES
C                                       OF THE COLS OF "PHI".
C
C     V1                                VECTOR CONTAINING THE NUMBER OF ONES
C                                       OF THE ROWS OF "PHI".
C
C     Z                                 DENOTES MEMBERS OF THE V' SET
C
C
C     ****NOTE****
C     THE DIMENSIONS OF "MIN", "V1", "V2", + "Z" SHOULD BE SET
C     EQUAL TO "SYS2".
C
C
      IMPLICIT INTEGER*2 (A-Z)
      LOGICAL*1 PHI(SYS2,SYS2), Z(256), FLAG(SYS2)
      DIMENSION MIN(256), V1(256), V2(256)
C
C     CALCULATE THE SET SET V
C
C     DETERMINE V (I) AND V (I) FOR Z(I)
C                1         2
C     WHERE: V (I) IS THE NUMBER OF 1'S IN COL I OF PHI
C             1
C            V (I) IS THE NUMBER OF 1'S IN ROW I OF PHI
C             2
C
      DO 1 I=1,DIMPHI
      V1(I) = 0
      V2(I) = 0
      IF(FLAG(I))  GOTO  1
      V1(I) = V1SET(PHI,I,DIMPHI,SYS2)
      V2(I) = V2SET(PHI,I,DIMPHI,SYS2)
    1 CONTINUE
C
C     NOW DETERMINE THE SET V'
C
C     V' = MAX2(MIN2(V1,V2))
C
C     FIND MIN2(V1,V2)
C
    2 DO 3 I=1,DIMPHI
      MIN(I) = 0
      IF(FLAG(I))  GOTO  3
      MIN(I) = MIN2(V1(I),V2(I))
    3 CONTINUE
C
C     GET MAX2(MIN2(V1,V2))
C
    4 BIGGER = MIN(1)
      ZPOINT = 0
      DO 5 I=1,DIMPHI
      IF(FLAG(I))  GOTO  5
      BIG = MAX2(BIGGER,MIN(I))
      IF(BIG .GT. BIGGER)  CALL ZER(Z,I,SYS2)
      IF(BIG .LE. MIN(I))    Z(I)     = .TRUE.
      IF(BIG .GE. BIGGER)  BIGGER = BIG
    5 CONTINUE
C
C     NOW FIND SET OF V' FOR WHICH
C     V1 + V2 IS A MAXIMUM
C
    9 BIGGER = 0
      DO 10 I=1,DIMPHI
      IF(.NOT. Z(I))  GOTO  10
      BIG = MAX2(V1(I) + V2(I),BIGGER)
      IF(BIG .LE. BIGGER)  GOTO  10
      BIGGER = BIG
      ZPOINT  = I
   10 CONTINUE
   11 RETURN
      END
      SUBROUTINE TBPHI(N,A,PHI,DIMPHI,SUBREL,SYS2,SYS)
C
C     THIS SUBROUTINE FORMS THE TRANSITIVE BORDERING INFERENCE
C     OPPORTUNITY MATRIX
C
C     WRITTEN BY: DAVID R. YINGLING, JR,
C                 ENGINEERING AND PUBLIC POLICY GROUP
C                 UNIVERSITY OF DAYTON
C                 DAYTON, OHIO 45469
C
C     VARIABLE NAME               DESCRIPTION
C
C     N                           THE NUMBER OF ELEMENTS IN "A",
C
C     A                           THE CURRENT MODEL MATRIX
C
C     PHI                         OUTPUT INFERENCE OPPORTUNITY MATRIX
C
C     DIMPHI                      THE CURRENT DIMENSION OF "PHI",
C
C     SUBREL                      SUBORDINATION RELATIONSHIP SWITCH,
C                                 IF THIS VALUE IS .TRUE., A SPECIAL
C                                 PROCEDURE IS PERFORMED ON THE "PHI"
C                                 MATRIX, IF .FALSE., NO SPECIAL
C                                 PROCEDURE IS DONE.
C
C     SYS2                        DIMENSION SIZES OF "PHI" AND
C                                 ASSOCIATED MATRICES
C
C     SYS                         DIMENSION SIZES OF SYSTEM MATRICES
C
C
      IMPLICIT INTEGER*2 (A-Z)
      INTEGER N
      LOGICAL SUBREL
      LOGICAL*1 A(SYS,SYS), PHI(SYS2,SYS2)
C
C
C     FORM THE INFERENCE OPPORTUNITY MATRIX PHI IN TWO STEPS
C
C     1. FORM N1
C
C             A       O
C
C    "N1" =
C
C             B       A
C
C                -T
C     WHERE: B = A   IF A TRANSITIVE RELATIONSHIP IS USED.
C
C                -T
C            B = A   + I IF A TRANSITIVE AND SUBORDINATION RELATIONSHIP
C            IS USED,
C
C     2. MULTIPLY ON "N1" TO OBTAIN PHI
C
C             A         O
```

```
C
C "PHI" =
C
C              ABA        A
C
C     DO STEP 1
C
      DO 1 I=1,N
      DO 1 J=1,N
C
C     PUT "A" ON UPPER LEFT OF "N1"
C
      PHI(I,J) = A(I,J)
C
C     PUT "A" ON LOWER RIGHT OF "N1"
C
      PHI(I + N,J + N) = A(I,J)
C
C     PUT "B" ON LOWER LEFT OF "N1"
C
      PHI(I + N,J) = .NOT. A(J,I)
    1 CONTINUE
C
C     SEE IF USER WANTS SUBORDINATION RELATION.
C     IF YES, ADD "I" TO "B" SECTION.
C     IF NO,  SKIP AROUND AND CONTINUE PROCESSING.
C
      IF(.NOT. SUBREL)  GOTO  2
      DO 3 I=1,N
      PHI(I + N,I) = .TRUE.
    3 CONTINUE
C
C     DO STEP 2
C
C     FIRST MULTIPLY  "B" TIMES "A" AND STORE IN "O" AS FIRST OPERATION
C
    2 DIM = N + 1
C
      DO 4 I=1,N
      DO 4 J=1,N
C
      PHI(I,J + N) = .FALSE.
C
      DO 4 K=1,N
      PHI(I,J + N) = PHI(I,J + N) .OR. (PHI(I+N,K) .AND. PHI(K+N,J+N))
    4 CONTINUE
C
C     NOW MULTIPLY "A" (LEFT OF "O") TIMES THE PRODUCT STORED IN
C     "O" AS SECOND OPERATION.  STORE THAT PRODUCT INTO ITS CORRECT
C     LOCATION (ABA).
C
      DO 5 I=1,N
      DO 5 J=1,N
C
      PHI(I + N,J) = .FALSE.
C
      DO 5 K=1,N
      PHI(I + N,J) = PHI(I + N,J) .OR. (PHI(I,K) .AND. PHI(K,J+N))
    5 CONTINUE
C
C     AS FINAL OPERATION, ZAP "0" SECTION BACK TO ZEROS
C
      DO 6 I=1,N
      DO 6 J=DIM,DIMPHI
      PHI(I,J) = .FALSE.
    6 CONTINUE
      RETURN
      END
      INTEGER FUNCTION VISET*2(PHI,J,DIMPHI,SYS2)
C
C     THIS FUNCTION SUBPROGRAM COMPUTES THE NUMBER OF
C     ONES IN COL "J" OF PHI.
C
C     WRITTEN BY:  DAVID R. YINGLING, JR
C                  ENGINEERING AND PUBLIC POLICY GROUP
C                  UNIVERSITY OF DAYTON
C                  DAYTON, OHIO  45469
C
      IMPLICIT INTEGER*2 (A-Z)
      LOGICAL*1 PHI(SYS2,SYS2)
      VISET = 0
      DO 1 I=1,DIMPHI
      IF(PHI(I,J))  VISET = VISET + 1
    1 CONTINUE
      RETURN
      END
      INTEGER FUNCTION V2SET*2(PHI,I,DIMPHI,SYS2)
C
C     THIS FUNCTION SUBPROGRAM COMPUTER THE NUMBER OF
C     ONES IN ROW "I" OF PHI.
C
C     WRITTEN BY:  DAVID R. YINGLING, JR
C                  ENGINEERING AND PUBLIC POLICY GROUP
C                  UNIVERSITY OF DAYTON
C                  DAYTON, OHIO  45469
C
      IMPLICIT INTEGER*2 (A-Z)
      LOGICAL*1 PHI(SYS2,SYS2)
      V2SET = 0
      DO 1 J=1,DIMPHI
      IF(PHI(I,J))  V2SET = V2SET + 1
    1 CONTINUE
      RETURN
      END
      SUBROUTINE DIGLEV(N,MATRIX,INDEX,TTYOUT,SYS)
C
C     THIS SUBROUTINE WILL DISPLAY THE DIGRAPH OF "MATRIX" IN A
C     LEVELS FORMAT
C
C     WRITTEN BY: DAVID R. YINGLING, JR.
C                 ENGINEERING AND PUBLIC POLICY GROUP
C                 UNIVERSITY OF DAYTON
C                 DAYTON, OHIO  45469
C
C     VARIABLE NAME                DESCRIPTION
C
C     N                            THE NUMBER OF ELEMENTS IN "MATRIX"
C
C     MATRIX                       REACHABILITY MATRIX TO BE DISPLAYED
C
C     INDEX                        THE INDEX SET OF "MATRIX"
C
C     TTYOUT                       FORTRAN WRITE UNIT NUMBER FOR
C                                  TELETYPE
C
C     SYS                          DIMENSION SIZES OF SYSTEM MATRICES
C
C--------
C
C     NC                           THE NUMBER OF ELEMENTS IN THE
C                                  CONDENSATION MATRIX
C
C     MATRXX                       SCRATCH MATRIX FOR LEVELS ROUTINE
C
C     INDXX                        SCRATCH INDEX VECTOR FOR LEVELS
C                                  ROUTINE
C
C     LEVELS                       SCRATCH VECTOR DENOTING THE NUMBER
C                                  OF ELEMENTS ON EACH LEVEL.
C                                  LEVELS(I)= NUMBER OF ELEMENTS ON
C                                  LEVEL #I.
C
C     NLEVEL                       THE TOTAL NUMBE OF LEVELS
C
C
C     ****NOTE****
C        THE DIMENSIONS OF "MATRXX", "INDXX", AND "LEVELS" SHOULD BE
C        EQUAL TO "SYS".
C
C
      IMPLICIT INTEGER*2 (A-Z)
      INTEGER N, INDEX, INDXX, NC
      DIMENSION INDEX(SYS), INDXX(128), LEVELS(128)
      LOGICAL*1 MATRIX(SYS,SYS),  MATRXX(128,128)
C
C     CHECK FOR ERROR
C
      IF(N .LE. 0)  GOTO  1
C
C     STEP 1---REARRANGE "MATRIX" INTO HIERARCHIAL FORM - PUT
C              RESULT IN "MATRXX" AND "INDXX".
C
```

```
      CALL HIERCH(N,MATRIX,INDEX,MATRXX,INDXX,NLEVEL,LEVELS,SYS)
C
C     STEP 2---PUT "MATRXX" INTO STANDARD FORM
C
      CALL STAN(N,MATRXX,INDXX,NLEVEL,LEVELS,SYS)
C
C     STEP 3---COMPUTE CONDENSATION MATRIX OF "MATRXX" - LEAVE
C              RESULT IN "MATRXX".
C
      NC = N
      CALL CONDE(NC,MATRXX,INDXX,LEVELS,TTYOUT,.TRUE.,SYS)
C
C     COMPUTE NONREDUNDANT ADJACENCY MATRIX (SKELETON MATRIX)
C
      CALL SKLTN(NC,MATRXX,SYS)
C
C     PRINT LEVELS FORMATTED DIGRAPH
C
      CALL DISPLV(NC,MATRXX,INDXX,LEVELS,TTYOUT,SYS)
      RETURN
C
C     ERROR MESSAGE
C
    1 WRITE(TTYOUT,100)
      RETURN
C
C     FORMAT
C
  100 FORMAT(42H-***ERROR*** NO STRUCTURE CURRENTLY EXISTS)
      END
      SUBROUTINE DISPLV(N,SKLTN,INDEX,LEVELS,TTYOUT,SYS)
C
C     THIS SUBROUTINE PRINTS A LEVEL FORMATTED DIGRAPH FROM
C     THE INPUT SKELETON MATRIX.
C
C     WRITTEN BY: DAVID R. YINGLING, JR.
C                 ENGINEERING AND PUBLIC POLICY GROUP
C                 UNIVERSITY OF DAYTON
C                 DAYTON, OHIO  45469
C
C     VARIABLE NAME                   DESCRIPTION
C
C     N                               NUMBER OF ELEMENTS IN THE INPUT
C                                     SKELETON MATRIX
C
C     SKLTN                           INPUT SKELETON MATRIX
C
C     INDEX                           INDEX SET OF "SKLTN".
C
C     LEVELS                          INPUT VECTOR DENOTING THE NUMBER
C                                     OF ELEMENTS ON EACH LEVEL.
C                                     LEVELS(I) = NUMBER OF ELEMENTS ON
C                                     LEVEL # I.
C
C     TTYOUT                          FORTRAN WRITE UNIT NUMBER FOR
C                                     TELETYPE.
C
C     SYS                             DIMENSION SIZES OF SYSTEM MATRICES
C
C-----------
C
C     LIST                            SCRATCH VECTOR FOR LEVELS PRINTOUT.
C                                     "LIST" CONTAINS THE ELEMENT NUMBERS
C                                     FOR PRINTING ON THE TELETYPE.
C
C     LEVEL                           THE CURRENT LEVEL NUMBER
C
C     ROW                             THE ROW OF THE LAST ELEMENT THAT
C                                     IS ON LEVEL # LEVEL.
C
C     COUNT                           THE NUMBER OF ELEMENTS IN "LIST".
C
C
C     ****NOTE****
C      THE DIMENSION OF "LIST" SHOULD BE EQUAL TO "SYS".
C
C
      IMPLICIT INTEGER*2 (A-Z)
      INTEGER N, INDEX
      DIMENSION INDEX(SYS), LEVELS(SYS), LIST(128)
      LOGICAL*1 SKLTN(SYS,SYS)
C
C     INITIALIZE PROCEDURE
C
      I     = 1
      LEVEL = 0
      ROW   = 1
C
```

```
C     PROCESS NEXT LEVEL
C
    1 LEVEL = LEVEL + 1
      ROW   = ROW + LEVELS(LEVEL)
      WRITE(TTYOUT,100)  LEVEL
C
C     IF THIS IS THE FIRST LEVEL, DO SPECIAL PROCESSING
C
    2 IF(LEVEL .EQ. 1)  GOTO  3
C
C     PROCESS ELEMENT #I........
C     FIND ALL ELEMENTS THAT ELEMENT #I REACHES TO
C
      COUNT = 0
      IMINUS= I - 1
      DO 4 J=1,IMINUS
      IF(.NOT. SKLTN(I,J))  GOTO  4
C
C     FOUND ONE, PUT INTO "LIST" FOR PRINTOUT
C
      COUNT        = COUNT + 1
      LIST(COUNT) = INDEX(J)
    4 CONTINUE
C
C     ALL DONE PROCESSING ELEMENT #I, PRINT OUT LINE
C
      WRITE(TTYOUT,101)  INDEX(I), (LIST(II),II=1,COUNT)
C
C     POINT TO NEXT ELEMENT
C
    5 I = I + 1
C
C     ARE WE DONE PRINTING THE DIGRAPH ???
C
      IF(I .GT. N)  RETURN
C
C     ARE WE DONE WITH THIS LEVEL ON DIGRAPH ???
C
      IF(I .EQ. ROW)  GOTO  1
      GOTO  2
C
C     SPECIAL PROCESSING FOR FIRST LEVEL
C
    3 WRITE(TTYOUT,102)  INDEX(I)
      GOTO  5
C
C     FORMATS
C
  100 FORMAT(1H-,10X,11HLEVEL NO.  ,I3/1H0)
  101 FORMAT(11X,I5,3H =>,10(7(I5,1H,)/20X))
  102 FORMAT(14X,I5)
      END
      SUBROUTINE HIERCH(N,INREA,INDXIN,REAH,INDXH,NL,LEVELS,SYS)
C
C     THIS SUBROUTINE REARRANGES A REACHABILITY MATRIX INTO A
C     LEVEL ORIENTED HIERARCHIAL REACHABILITY MATRIX.
C
C
C     VARIABLE NAME                 DESCRIPTION
C
C     INDXIN                        VECTOR CONTAINING THE INDEX SET OF
C                                   THE INPUT REACHABILITY MATRIX,
C
C     INDXH                         VECTOR CONTAINING THE INDEX SET OF THE
C                                   OUTPUT HIERARCHIAL REACHABILITY MATRIX
C
C     FLAG                          LOGICAL VECTOR WHICH DENOTES ELEMENTS
C                                   THAT HAVE ALREADY BEEN PROCESSED.  IF
C                                   FLAG(I) = .TRUE., ELEMENT #I HAS BEEN
C                                   PROCESSED.
C
C     LEVELS                        VECTOR CONTAINING THE NUMBER OF
C                                   ELEMENTS ON EACH LEVEL.  LEVELS(I) =
C                                   NUMBER OF ELEMENTS ON LEVEL #I.
C
C     TEMP                          SCRATCH VECTOR USED BY LEVELS
C                                   PARTITION ALGORITHM,. IT HOLDS THE
C                                   ELEMENTS THAT ARE ON THE CURRENT LEVEL
C
C     INREA                         INPUT (ARGUMENT) REACHABILITY MATRIX
C
C     REAH                          OUTPUT (RESULTANT) HIERARCHIAL
C                                   REACHABILITY MATRIX
C
C     N                             NUMBER OF ELEMENTS IN BOTH INPUT
C                                   REACHABILITY MATRIX AND OUTPUT
C                                   HIERARCHIAL REACHABILITY MATRIX
C
C     NEAP                          NUMBER OF ELEMENTS ALREADY PROCESSED
C
C     NEL                           NUMBER OF ELEMENTS ON CURRENT LEVEL
C
C     NL                            NUMBER OF CURRENT LEVEL
C
C     SYS                           DIMENSION SIZES OF SYSTEM MATRICES
C
C     ****NOTE****
C            THE DIMENSIONS OF "TEMP" AND "FLAG" SHOULD BE EQUAL TO
C            "SYS".
C
      IMPLICIT INTEGER*2 (A-Z)
      INTEGER N, INDXIN, INDXH
      DIMENSION INDXIN(SYS), INDXH(SYS), LEVELS(SYS), TEMP(128)
      LOGICAL*1 INREA(SYS,SYS), REAH(SYS,SYS), FLAG(128)
C
C     COPY INREA INTO REAH : INITIALIZE FLAG
C
      DO 1 I=1,N
      FLAG(I) = .FALSE.
      DO 1 J=1,N
      REAH(I,J) = INREA(I,J)
    1 CONTINUE
C
C     INITIALIZE LEVELS PARTITION ALGORITHM
C
      NL   = 0
      NEAP = 0
C
C     BEGIN LEVELS PARTITION ALGORITHM
C
C     THIS ALGORITHM REARRANGES INDXIN ACCORDING TO A LEVELS
C     PARTITION.  THE RESULT IS IN INDXH.
C
    2 NL = NL + 1
      NEL= 0
C
C     FIND AN ELEMENT TO PROCESS
C
      DO 3 I=1,N
      IF(FLAG(I))  GOTO  3
C
C     TEST TO SEE IF THE REACHABILITY SET (R) IS A
C     SUBSET OF THE ANTECEEDENT SET (A) FOR THIS ELEMENT
C
      DO 4 J=1,N
      IF(REAH(I,J) .AND. .NOT. REAH(J,I))  GOTO  3
    4 CONTINUE
C
C     COME HERE IF R WAS A SUBSET OF A
C
      NEL = NEL + 1
      INDXH(NEAP + NEL) = INDXIN(I)
      TEMP(NEL) = I
    3 CONTINUE
      NEAP       = NEAP + NEL
      LEVELS(NL) = NEL
C
C     FOUND ALL ELEMENTS ON CURRENT LEVEL (NL).
C     BLANK ROW AND COL ON REAH FOR ALL ELEMENTS ON THIS LEVEL.
C
      DO 5 I=1,NEL
      TEMPI = TEMP(I)
      FLAG(TEMPI) = .TRUE.
      DO 5 J=1,N
      REAH(TEMPI,J) = .FALSE.
      REAH(J,TEMPI) = .FALSE.
    5 CONTINUE
C
C     CHECK TO SEE IF ALL ELEMENTS HAVE BEEN PROCESSED
C
      IF(NEAP .LT. N)  GOTO  2
C
C     COME HERE WHEN ALL ELEMENTS HAVE BEEN PROCESSED.
C     LEVELS PARTITION ALGORITHM IS NOW COMPLETE, BEGIN TO
C     CONSTRUCT A LEVELS PARTITIONED HIERARCHIAL REACHABILITY
C     MATRIX BASED ON INDXH.
C
C     EXCHANGE ALL ROWS FIRST ACCORDING TO INDXH
```

```
C
      DO 6 I=1,N
      DO 7 J=1,N
      IF(INDXH(I) .EQ. INDXIN(J))  GOTO  8
    7 CONTINUE
C
    8 DO 6 K=1,N
      REAH(I,K) = INREA(J,K)
    6 CONTINUE
C
C     COPY REAH INTO INREA.  THIS IS A NECESSARY STEP, DO NOT
C     TAKE OUT !!!!!
C
      DO 9 I=1,N
      DO 9 J=1,N
      INREA(I,J) = REAH(I,J)
    9 CONTINUE
C
C     WE PRESENTLY HAVE A MATRIX (INREA) WHICH IS INDEXED ON THE
C     TOP BY INDXH AND DOWN THE SIDE BY INDXIN.  REARRANGE THE
C     COLS SO THEY ALSO ARE INDEXED BY INDXH.  LEAVE RESULT IN
C     REAH.
C
      DO 10 I=1,N
      DO 11 J=1,N
      IF(INDXH(I) .EQ. INDXIN(J))  GOTO  13
   11 CONTINUE
C
   13 DO 10 K=1,N
      REAH(K,I) = INREA(K,J)
   10 CONTINUE
C
C     IN ITS PRESENT FORM, INREA, FOR ALL PURPOSES OTHER THAN
C     THIS SUBROUTINE, IS SCRAMBLED.  COPY REAH INTO INREA TO
C     SOLVE THIS PROBLEM.
C
      DO 14 I=1,N
      INDXIN(I) = INDXH(I)
      DO 14 J=1,N
      INREA(I,J) = REAH(I,J)
   14 CONTINUE
C
      RETURN
      END
      SUBROUTINE STAN(N,MATRIX,INDEX,NLEVEL,LEVELS,SYS)
C
C     THIS SUBROUTINE CONVERTS AN INPUT HIERARCHIAL REACHABILITY
C     MATRIX (MATRIX) INTO ITS STANDARD FORM.
C
C     EDITED BY: DAVID R. YINGLING, JR.
C                ENGINEERING AND PUBLIC POLICY GROUP
C                UNIVERSITY OF DAYTON
C                DAYTON, OHIO  45469
C
C
C     VARIABLE NAME                    DESCRIPTION
C
C     N                                NUMBER OF ELEMENTS IN INPUT MATRIX
C
C     MATRIX                           INPUT/OUTPUT HIERARCHIAL REACHABILITY
C                                      MATRIX.
C
C     INDEX                            INPUT/OUTPUT INDEX SET OF "MATRIX".
C
C     LEVELS                           INPUT VECTOR DENOTING THE NUMBER OF
C                                      ELEMENTS ON EACH LEVEL.  LEVELS(I) =
C                                      THE NUMBER OF ELEMENTS ON LEVEL #I
C
C     NLEVEL                           THE TOTAL NUMBER OF LEVELS
C
C     SYS                              DIMENSION SIZES OF SYSTEM MATRICES
C
C-----------
C
C     NONES                            NUMBER OF ONES COUNTED
C
C     END                              ENDING SUBSCRIPT FOR LEVEL #I
C
C     START                            STARTING SUBSCRIPT FOR LEVEL #I
C
C
      IMPLICIT INTEGER*2 (A-Z)
      INTEGER N, INDEX
      DIMENSION INDEX(SYS), LEVELS(SYS)
      LOGICAL*1 MATRIX(SYS,SYS), SWIT
C
C     CHECK FIRST TO MAKE SURE ALL NON-CYCLE ELEMENTS
C     ARE UPPERMOST ON EACH LEVEL
C
      END = 0
      DO 1 I=1,NLEVEL
      START = END + 1
      END   = END + LEVELS(I)
C
C     IF THE NUMBER OF ELEMENTS ON LEVEL #I IS TWO OR
C     LESS, NO RE-ADJUSTMENT IS NECESSARY
C
      IF(LEVELS(I) .LE. 2)  GOTO  1
C
C     FIND AND MOVE NON-CYCLE ELEMENTS UP ON MATRIX IF NECESSARY
C
      DO 1 II=START,END
      LAST = 0
      DO 1 ROW=START,END
      NONES = 0
C
      DO 2 COL=START,END
      IF(MATRIX(ROW,COL))  NONES = NONES + 1
    2 CONTINUE
C
C     CHECK TO SEE IF ELEMENT #ROW HAS LESS ONES THAN
C     LAST ELEMENT CHECKED
C
      IF(NONES .LT. LAST)  CALL SWITCH(N,MATRIX,INDEX,ROW,SYS)
      IF(NONES .GE. LAST)  LAST = NONES
    1 CONTINUE
C
C     ALL NON-CYCLE ELEMENTS ARE AT THE BEGINING OF LEVELS
C     PARTITION.  NOW GROUP THE CYCLES TOGETHER.
C
      NMINUS = N - 1
    3 SWIT = .FALSE.
C
C     CHECK FOR ONES ABOVE MAIN DIAGONAL
C
      DO 4 I=1,NMINUS
      IPLUS = I + 1
      DO 5 J=IPLUS,N
      IF(.NOT. MATRIX(I,J))  GOTO  5
C
C     COME HERE IF A ONE ABOVE THE MAIN DIAGONAL IS FOUND
C
C     CHECK TO SEE IF IT IS NEXT TO DIAGONAL ONE--IF SO,
C     DON'T SWITCH BECAUSE OF THE WAY THE "SWITCH" SUBROUTINE
C     WORKS.  IF NOT, SWITCH THAT ELEMENT (J) WITH ELEMENT J-1.
C
    6 JMINUS = J - 1
      IF(I .EQ. JMINUS)  GOTO  4
C
      CALL SWITCH(N,MATRIX,INDEX,J,SYS)
      SWIT = .TRUE.
      J = J - 1
      GOTO  6
    5 CONTINUE
    4 CONTINUE
C
C     IF ANY SWITCHING WAS DONE, WE NEED TO CHECK AGAIN
C     OTHERWISE RETURN.
C
      IF(SWIT)  GOTO  3
      RETURN
      END
      SUBROUTINE SWITCH(N,MATRIX,INDEX,ROW,SYS)
C
C     THIS SUBROUTINE WILL SWITCH THE ROW AND COL OF "ROW" WITH
C     THE ROW AND COL OF "ROW" - 1.
C
C     EDITED BY:  DAVID R. YINGLING, JR.
C                 ENGINEERING AND PUBLIC POLICY GROUP
C                 UNIVERSITY OF DAYTON
C                 DAYTON, OHIO  45469
C
C
C     VARIABLE NAME                DESCRIPTION
C
C     N                            THE TOTAL NUMBER OF ELEMENTS IN
C                                  "MATRIX".
C
C     MATRIX                       INPUT MATRIX TO BE SWITCHED
C
C     INDEX                        INDEX SET OF THE INPUT MATRIX
C
C     ROW                          THE SUBSCRIPT OF THE MATRIX TO BE
C                                  SWITCHED.
C
C     SYS                          DIMENSION SIZE OF SYSTEM MATRICES
C
C-----
C
C     OTHER                        THE OTHER ROW TO BE SWITCHED. ALWAYS
C                                  EQUAL TO "ROW" - 1
C
      IMPLICIT INTEGER*2 (A-Z)
      INTEGER N, INDEX, ITEMP
      DIMENSION INDEX(SYS)
      LOGICAL*1 MATRIX(SYS,SYS), TEMP
C
      OTHER = ROW - 1
C
C     SWITCH THE ROWS
C
      DO 1 I=1,N
      TEMP            = MATRIX(ROW,I)
      MATRIX(ROW,I)   = MATRIX(OTHER,I)
      MATRIX(OTHER,I) = TEMP
    1 CONTINUE
C
C     SWITCH THE COLS
C
      DO 2 I=1,N
      TEMP            = MATRIX(I,ROW)
      MATRIX(I,ROW)   = MATRIX(I,OTHER)
      MATRIX(I,OTHER) = TEMP
    2 CONTINUE
C
C     SWITCH THE INDEX SET
C
      ITEMP        = INDEX(ROW)
      INDEX(ROW)   = INDEX(OTHER)
      INDEX(OTHER) = ITEMP
```

```
C
      RETURN
      END
      SUBROUTINE CUNDE(N,MATRIX,INDEX,LEVELS,TTYOUT,TYPE,SYS)
C
C     THIS SUBROUTINE TAKES THE INPUT HIERARCHIAL REACHABILITY MATRIX
C     IN STANDARD FORM AND REDUCES EACH MAXIMAL CYCLE SET INTO A
C     SINGLE PROXY ELEMENT -- THEREBY FORMING THE CONDENSATION MATRIX.
C
C     EDITED BY:  DAVID R. VINGLING, JR.
C                 ENGINEERING AND PUBLIC POLICY GROUP
C                 UNIVERSITY OF DAYTON
C                 DAYTON, OHIO  45469
C
C     VARIABLE NAME                 DESCRIPTION
C
C     N                             THE NUMBER OF ELEMENTS IN THE INPUT
C                                   MATRIX.  UPON COMPLETION OF THIS
C                                   SUBROUTINE, THE NEW VALUE OF N WILL
C                                   REFLECT THE ELEMENTS DELETED.
C
C     MATRIX                        THE INPUT STANDARD FORM MATRIX/OUTPUT
C                                   CONDENSATION MATRIX
C
C     INDEX                         THE INDEX SET OF "MATRIX".
C
C     LEVELS                        INPUT VECTOR WHICH CONTAINS THE
C                                   NUMBER OF ELEMENTS ON EACH LEVEL.
C                                   LEVELS(I) = NUMBER OF ELEMENTS ON
C                                   LEVEL #I.
C
C     TTYOUT                        FORTRAN WRITE UNIT NUMBER FOR
C                                   THE TELETYPE
C
C     TYPE                          LOGICAL VARIABLE WHICH WHEN .FALSE.
C                                   SUPRESSES THE PRINTING OF CYCLES AT
C                                   THE TELETYPE. WHEN.TRUE., CYCLES
C                                   ARE PRINTED
C
C     SYS                           DIMENSION OF SYSTEM MATRICES
C
C--------------
C
C     LIST                          A SCRATCH VECTOR OF DIMENSION "SYS"
C                                   THAT HOLDS THE INDEX NUMBERS OF THE
C                                   CYCLE SET BEING OPERATED ON.
C
C     COUNT                         THE NUMBER OF ELEMENTS IN THE CURRENT
C                                   CYCLE SET BEING OPERATED ON.
C
C     POSITN                        THE POSITION TALLY IN THE "LEVELS"
C                                   VECTOR.
C
C     *****NOTE*****
C            THE DIMENSION OF "LIST" SHOULD BE EQUAL TO "SYS".
C
      IMPLICIT INTEGER*2 (A-Z)
      INTEGER N, INDEX
      DIMENSION INDEX(SYS), LEVELS(SYS), LIST(128)
      LOGICAL   TYPE
      LOGICAL*1 MATRIX(SYS,SYS)
```

```
C
C     CHECK FOR A ONE ABOVE THE MAIN DIAGONAL (FIND A CYCLE)
C
C
      I          = 1
    1 COUNT      = 1
      LIST(1) =  INDEX(I)
      J          = I + 1
      IF(.NOT. MATRIX(I,J))  GOTO  5
C
C     A ONE WAS FOUND, PUT THAT ELEMENT INTO CYCLE PRINTOUT LIST
C     AND THEN ELIMINATE FROM THE MATRIX.
C
    2 COUNT          = COUNT + 1
      LIST(COUNT) = INDEX(J)
      CALL ELIM(N,MATRIX,INDEX,J,SYS)
C
C     NOW REDUCE NUMBER OF ELEMENTS ON LEVEL WHERE AN ELEMENT
C     WAS JUST ELIMINATED.
C
      POSITN = 0
      DO 3 II=1,N
      POSITN = POSITN + LEVELS(II)
      IF(POSITN .GE. J)  GOTO  4
    3 CONTINUE
C
    4 LEVELS(II) = LEVELS(II) - 1
C
C     ANY MORE ELEMENTS IN THIS CYCLE SET???
C
      IF(MATRIX(I,J) .AND. J .LE. N)  GOTO  2
C
C     WRITE CYCLE OUT TO TELETYPE
C
      IF (TYPE) WRITE(TTYOUT,100)  (LIST(III), III=1,COUNT)
    5 I = I + 1
      IF(I .LT. N)  GOTO  1
      RETURN
C
C     FORMAT
C
  100 FORMAT(11H0  CYCLE ON,2X,13(10(I4,1H,)/13X))
      END
      SUBROUTINE ELIM(N,MATRIX,INDEX,DELETE,SYS)
C
C     THIS SUBROUTINE ELIMINATES AN ELEMENT FROM A GIVEN
C     INPUT MATRIX.
C
C     EDITED BY: DAVID R. YINGLING, JR.
C                ENGINEERING AND PUBLIC POLICY GROUP
C                UNIVERSITY OF DAYTON
C                DAYTON, OHIO   45469
C
C     VARIABLE NAME                 DESCRIPTION
C
C     N                             NUMBER OF ELEMENTS IN "MATRIX", UPON
C                                   COMPLETION OF THIS ROUTINE, THE NEW
C                                   VALUE OF "N" WILL REFLECT THE ELEMENT
C                                   DELETED.
C
C     MATRIX                        THE INPUT/OUTPUT MATRIX
C
C     INDEX                         THE INDEX SET OF "MATRIX".
C
C     DELETE                        THE SUBSCRIPT OF "MATRIX" TO BE
C                                   ELIMINATED.
C
C     SYS                           DIMENSION SIZE OF SYSTEM MATRICES
C
      IMPLICIT INTEGER*2 (A-Z)
      INTEGER N, INDEX
      DIMENSION INDEX(SYS)
      LOGICAL*1 MATRIX(SYS,SYS)
C
C     CHECK FOR "DELETE" .EQ. TO LAST LOGICAL POSITION ON "MATRIX"
C
      NMINUS = N - 1
      IF(DELETE .EQ. N)  GOTO  1
C
C     MOVE ALL COLUMNS BELOW "DELETE" OVER BY 1
C
      DO 2 ROW1=DELETE,NMINUS
      ROW2 = ROW1 + 1
C
      DO 3 COL=1,N
      MATRIX(ROW1,COL) = MATRIX(ROW2,COL)
    3 CONTINUE
C
C     MOVE ALL ROWS BELOW "DELETE" UP BY 1
C
      DO 2 ROW=1,N
      MATRIX(ROW,ROW1) = MATRIX(ROW,ROW2)
    2 CONTINUE
C
C     FIX UP INDEX SET
C
      DO 4 ROW1=DELETE,NMINUS
      ROW2 = ROW1 + 1
      INDEX(ROW1) = INDEX(ROW2)
    4 CONTINUE
C
```

```
C     SUBTRACT ONE FROM "N" TO REFLECT DELETION
C
    1 N = NMINUS
      RETURN
      END
      SUBROUTINE SKLTN(N,MATRIX,SYS)
C
C     THIS SUBROUTINE CONVERTS THE INPUT MATRIX INTO A
C     NONREDUNDANT ADJACENCY MATRIX (SKELETON MATRIX)
C
C        THIS ALGORITHM IS SIMILAR TO THE ONE DESCRIBED BY
C        R.K. SHYAMASUNDAR, "BOOLEAN MATRIX METHOD FOR THE
C        CONSTRUCTION OF HIERARCHIAL GRAPHS", IEEE TRANSACTIONS
C        ON SYSTEMS, MAN, AND CYBERNETICS, VOL.SMC-8, NO. 2,
C        FEBRUARY, 1978.
C
C     EDITED BY: DAVID R. YINGLING, JR.
C                ENGINEERING AND PUBLIC POLICY GROUP
C                UNIVERSITY OF DAYTON
C                DAYTON, OHIO  45469
C
C     VARIABLE NAME                 DESCRIPTION
C
C     N                             NUMBER OF ELEMENTS IN "MATRIX".
C
C     MATRIX                        INPUT/OUTPUT MATRIX TO BE CONVERTED
C
C     SYS                           DIMENSION SIZES OF SYSTEM MATRICES
C
      IMPLICIT INTEGER*2 (A-Z)
      INTEGER N
      LOGICAL*1 MATRIX(SYS,SYS)
C
      NMINUS = N - 1
      DO 1 I=2,NMINUS
      IMINUS = I - 1
      DO 1 J=1,IMINUS
C
C     CHECK REACHABILITY OF NODE J TO NODE I
C
      IF(.NOT. MATRIX(I,J))  GOTO  1
C
C     ADD ALL NODES TO ROW J THAT CAN BE REACHED FROM NODE I
C
      IPLUS = I + 1
      DO 1 K=IPLUS,N
      MATRIX(K,J) = MATRIX(K,J) .AND. .NOT. MATRIX(K,I)
    1 CONTINUE
      RETURN
      END
      SUBROUTINE ZER(VECTOR,I,SYS2)
C
C     THIS SUBROUTINE ZEROS OUT ALL PREVIOUSLY
C     FLAGED MAXIMUMS
C
C     WRITTEN BY:  DAVID R. YINGLING, JR
C                  ENGINEERING AND PUBLIC POLICY GROUP
C                  UNIVERSITY OF DAYTON
C                  DAYTON, OHIO  45469
C
      IMPLICIT INTEGER*2 (A-Z)
      LOGICAL*1  VECTOR(SYS2)
C
      II = I - 1
      DO 1 J=1,II
      VECTOR(J) = .FALSE.
    1 CONTINUE
      RETURN
      END
      SUBROUTINE QUEST(EL1,EL2,TTYOUT,QTYPE,TXTWDS)
C
C     THIS SUBROUTINE DISPLAYS EITHER FULL TEXT
C     OR SYMBOLIC QUERIES
C
C     WRITTEN BY:  DAVID R. YINGLING, JR
C                  ENGINEERING AND PUBLIC POLICY GROUP
C                  UNIVERSITY OF DAYTON
C                  DAYTON, OHIO  45469
C
      IMPLICIT INTEGER (A-Z)
      INTEGER*2 EL1,EL2,TTYOUT,TXTWDS
      LOGICAL QTYPE
      DIMENSION BLOCK(800),BUFFER(256)
      EQUIVALENCE (BLOCK,R1)
      COMMON /FTEXT/  R1(160), L1(160), R2(160), L2(160), R3(160)
      COMMON UNUSED
      DATA CRLF/Z15152551/, INIT/Z1525250C/
C
C     ****NOTE****
C           THE DIMENSION OF "R1", "L1", "R2", "L2", "R3" SHOULD BE
C           EQUAL TO "TXTWDS".
C
C           THE DIMENSION OF "BLOCK" SHOULD BE EQUAL TO "TXTWDS * 5.0".
C
C
C     SYMBOLIC QUERIES ?
C
      IF(QTYPE)   GOTO  1
C
C     NOPE, FULL TEXT - READ IN ELEMENTS
C
      I1 = EL1 + 4
      FIND(8'I1)
      I2 = EL2 + 4
      READ(8'I1)  (L1(I),I=1,TXTWDS)
      READ(8'I2)  (L2(I),I=1,TXTWDS)
      OFFSET = 0
C
C     PRESENT FIVE LINES OF I/O
C       1) INTRODUCTORY CLAUSE   (RELATIONAL CLAUSE 1)
C       2) ELEMENT A
C       3) CORRALATION CLAUSE   (RELATIONAL CLAUSE 2)
C       4) ELEMENT B
C       5) QUALIFYING CLAUSE   (RELATIONAL CLAUSE 3)
C
      COUNT = 8
      I4= 2
      BUFFER(I4) = INIT
      DO 2 I= 1,5
      I1 = 0
      I2 = 0
C
C     PRINT UP TO TEN LINES FOR EACH OF THE ABOVE PHRASES
C
      DO 3 J = 1,10
C
C     IF LENGTH INDICATOR IS ZERO, DON'T PRINT
```

```
C
      IF(BLOCK(J+OFFSET) .EQ. 0)   GOTO  3
C
C     NOT ZERO, COMPUTE LENGTH AND LOCATION OF LINE
C
      LENGTH = BLOCK(J+OFFSET)
      I1 = I2 + 1
      I2 = I1 + LENGTH - 1
C
C     PUT TEXT INTO BUFFER
C
      I3 = I4 + 1
      I4 = I3 + LENGTH - 1
      C  = I1
C
      DO 4 III=I3,I4
      BUFFER(III) = BLOCK(C + OFFSET + 10)
      C = C + 1
    4 CONTINUE
C
      I4 = I4 + 1
      BUFFER(I4) = CRLF
      COUNT       = COUNT + (LENGTH * 4) + 4
    3 CONTINUE
      OFFSET = OFFSET + TXTWDS
    2 CONTINUE
      COUNT = COUNT - 4
      CALL ZAP(COUNT,BUFFER)
      GOTO  6
C
C     SYMBOLIC TEXT QUERIES
C
    1 WRITE(TTYOUT,101)   EL1,EL2
    6 RETURN
C
C     CHECK 4 FORMAT
C
  100 FORMAT(1X,17A4)
  101 FORMAT(1H-,I5,2H R,I5,2H? )
      END
      SUBROUTINE FINDIT(N,N1,N2,S1,S2,INDEX,FOUND1,FOUND2,SYS)
C
C     THIS SUBROUTINE FINDS ELEMENTS N1, N2 IN THE INDEX SET
C
C     THE VALUES S1, S2 ARE THE POSITIONS OF N1 AND N2 IN THE
C     INDEX SET.
C
C     FOUND1 AND FOUND2 ARE LOGICAL VALUES AND ARE SET EQUAL
C     TO .TRUE. IF N1 OR N2 (RESPECTIVELY) ARE FOUND IN THE
C     INDEX SET.
C
C     WRITTEN BY:  DAVID R. VINGLING, JR.
C                  ENGINEERING AND PUBLIC POLICY GROUP
C                  UNIVERSITY OF DAYTON
C                  DAYTON, OHIO  45469
C
      IMPLICIT INTEGER*2 (A-Z)
      INTEGER N, INDEX
      DIMENSION INDEX(SYS)
      LOGICAL FOUND1, FOUND2
C
      FOUND1 = .FALSE.
      FOUND2 = .FALSE.
      S1     = 0
      S2     = 0
C
      DO 1 I=1,N
      IF(N1 .EQ. INDEX(I))  S1 = I
      IF(N2 .EQ. INDEX(I))  S2 = I
    1 CONTINUE
C
      IF(S1 .GT. 0)  FOUND1 = .TRUE.
      IF(S2 .GT. 0)  FOUND2 = .TRUE.
      RETURN
      END
      SUBROUTINE GETNUM(ARRAY,N,TTYIN,TTYOUT)
C
C     THIS SUBROUTINE WILL READ "N" UNSIGNED INTEGERS FROM
C     THE TERMINAL TYPED IN A FREE FORMAT AND STORE THEM IN
C     "ARRAY"
C
C     WRITTEN BY:  DAVID R. YINGLING, JR.
C                  ENGINEERING AND PUBLIC POLICY GROUP
C                  UNIVERSITY OF DAYTON
C                  DAYTON, OHIO  45469
C
      IMPLICIT INTEGER*2 (A-Z)
      INTEGER NUMS,BUFFER,BLANK,COMMA
      DIMENSION ARRAY(1), BUFFER(80), NUMS(10)
      DATA NUMS/1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9/
      DATA BLANK/1H /, COMMA/1H,/
C
C
    1 READ(TTYIN,200)  BUFFER
C
      L        = N
      ARRAY(L) = 0
      POWER    = 0
C
      DO 2 I=1,80
      K = 81 - I
      IF(BUFFER(K) .EQ. BLANK .OR. BUFFER(K) .EQ. COMMA)  GOTO  3
C
C     FOUND A CHARACTER, SEE IF IT'S A VALID NUMERIC
C
      DO 4 J=1,10
      II = J - 1
      IF(BUFFER(K) .NE. NUMS(J))  GOTO  4
C
C     ITS A NUMBER, ADD IT TO PRESENT SUM
C
      ARRAY(L) = ARRAY(L) + (II * (10**POWER))
      POWER    = POWER + 1
      GOTO  2
    4 CONTINUE
C
C     COME HERE IF CHARACTER FOUND WAS NOT NUMERIC
C
      WRITE(TTYOUT,100)
      GOTO  1
C
C     FOUND A DELIMITER, SEE IF END OF A NUMBER
C
    3 IF(ARRAY(L) .EQ. 0)  GOTO  2
      L = L - 1
      IF(L .EQ. 0)  GOTO  5
      ARRAY(L) = 0
      POWER    = 0
    2 CONTINUE
C
C     MAKE SURE NUMBER(S) IS/ARE LESS THAN 99999 SO
C     WE DON'T EXCEED I5 FORMATS
C
```

```
    5 DO 6 I=1,N
      IF(ARRAY(I) .GT. 99999)  GOTO  7
    6 CONTINUE
      RETURN
C
C     ERROR MESSAGE
C
    7 WRITE(TTYOUT,101)
      GOTO  1
C
C     FORMATS
C
  100 FORMAT(37H-***ERROR*** INPUT NOT NUMERIC--RETRY)
  101 FORMAT(39H-***ERROR*** NUMBER(S) TOO LARGE--RETRY)
  200 FORMAT(80A1)
      END
      SUBROUTINE IU(N,MAT,INDEX,UNITNO,READ,SYS)
C
C     THIS SUBROUTINE HANDLES ALL PERMFIL I/O
C
C     IT WRITES/READS N (THE NUMBER OF ELEMENTS IN THE MATRIX),
C     THE SYSTEM MATRIX (MAT), AND THE INDEX SET (INDEX) TO/FROM
C     A PERMFILE.
C
C     WRITTEN BY:  DAVID R. YINGLING, JR
C                  ENGINEERING AND PUBLIC POLICY GROUP
C                  UNIVERSITY OF DAYTON
C                  DAYTON, OHIO  45469
C
      IMPLICIT INTEGER*2 (A-Z)
      INTEGER N, INDEX
      LOGICAL READ
      LOGICAL*1 MAT(SYS,SYS)
      DIMENSION INDEX(SYS)
C
C     CHECK TO SEE IF THIS IS A READ REQUEST
C
      IF(READ)  GOTO  1
C
C     NUP, IT'S A WRITE REQUEST
C
      REWIND UNITNO
      WRITE(UNITNO) N
      WRITE(UNITNO) MAT
      WRITE(UNITNO) INDEX
      GOTO  2
C
C     YUP, IT WAS A READ REQUEST
C
    1 REWIND UNITNO
      READ (UNITNO) N
      READ (UNITNO) MAT
      READ (UNITNO) INDEX
C
    2 RETURN
      END
      SUBROUTINE DIGSTG(N,MATRIX,INDEX,TTYOUT,SYS)
C
C     THIS SUBROUTINE DISPLAYS THE DIGRAPH IN STAGES
C
C     WRITTEN BY:  DAVID R. VINGLING, JR.
C                  ENGINEERING AND PUBLIC POLICY GROUP
C                  UNIVERSITY OF DAYTON
C                  DAYTON, OHIO  45469
C
C
C     VARIABLE NAME               DESCRIPTION
C
C     N                           THE NUMBER OF ELEMENTS IN THE INPUT
C                                 MATRIX.
C
C     MATRIX                      INPUT REACHABILITY MATRIX
C
C     INDEX                       INDEX SET OF THE INPUT MATRIX
C
C     TTYOUT                      FORTRAN WRITE UNIT NUMBER FOR TELETYPE
C
C     SYS                         DIMENSION SIZES OF SYSTEM MATRICES
C
C-----------
C
C     MATRXX                      SCRATCH MATRIX FOR STAGES ROUTINE
C
C     INDXX                       INDEX SET OF SCRATCH MATRIX
C
C     STAGES                      VECTOR DENOTING THE NUMBER OF ELEMENTS
C                                 ON EACH STAGE.  STAGES(I) = THE
C                                 NUMBER OF ELEMENTS UN STAGE #I.
C
C     NS                          THE TOTAL NUMBER OF STAGES
C
C     ****NOTE****
C          THE DIMENSIONS OF "MATRXX", "INDXX", + "STAGES" SHOULD BE
C          EQUAL TO "SYS".
C
C     ****NOTE****
C          IN ORDER TO CONSERVE CORE STORAGE, THE STATE OF THE INPUT
C          REACHABILITY MATRIX HAS BEEN DESTROYED.  LET THE PROGRAMMER
C          BEWARE!!!!!!!!
C
      IMPLICIT INTEGER*2 (A-Z)
      INTEGER N, INDEX, INDXX
      DIMENSION INDEX(SYS), INDXX(128), STAGES(128)
      LOGICAL*1 MATRIX(SYS,SYS), MATRXX(128,128)
C
C     CHECK FOR ERROR CONDITION
C
      IF(N .LE. 0)  GOTO  4
C
C     STEP 1---TRANSPOSE INPUT ORIGINAL REACHABILITY MATRIX
C
      DO 1 I=1,N
      DO 1 J=1,N
      MATRXX(I,J) = MATRIX(J,I)
```

```
    1 CONTINUE
C
      DO 2 I=1,N
      DO 2 J=1,N
      MATRIX(I,J) = MATRXX(I,J)
    2 CONTINUE
C
C     STEP 2---LEVELS PARTITION "MATRIX", LEAVE RESULT IN "MATRXX"
C
      CALL HIERCH(N,MATRIX,INDEX,MATRXX,INDXX,NS,STAGES,SYS)
C
C     STEP 3---PUT "MATRXX" INTO STANDARD FORM
C
      CALL STAN(N,MATRXX,INDXX,NS,STAGES,SYS)
C
C     STEP 4-- CALCULATE CONDENSATION MATRIX
C
      CALL CONDE(N,MATRXX,INDXX,STAGES,TTYOUT,.TRUE.,SYS)
C
C     STEP 5---CALCULATE SKELETON MATRIX
C
      CALL SKLTN(N,MATRXX,SYS)
C
C     STEP 6---TRANSPOSE SKELETON MATRIX  TO PUT INTO UPPER
C     TRIANGULAR FORM
C
      DO 3 I=1,N
      DO 3 J=1,N
      MATRIX(I,J) = MATRXX(J,I)
    3 CONTINUE
C
C     STEP 7---PRINT OUT STAGE DIGRAPH
C
      CALL DISPST(N,MATRIX,INDXX,STAGES,NS,TTYOUT,SYS)
      RETURN
C
C     ERROR MESSAGE
C
    4 WRITE(TTYOUT,100)
      RETURN
C
C     FORMAT
C
  100 FORMAT(42H-***ERROR*** NO STRUCTURE CURRENTLY EXISTS)
      END
      SUBROUTINE DISPST(N,MATRIX,INDEX,STAGES,NSTAGE,TTYOUT,SYS)
C
C     THIS SUBROUTINE TAKES AN UPPER TRIANGULAR SKELETON MATRIX
C     AND PRINTS OUT A STAGES DIGRAPH
C
C     WRITTEN BY:  DAVID R. YINGLING, JR.
C                  ENGINEERING AND PUBLIC POLICY GROUP
C                  UNIVERSITY OF DAYTON
C                  DAYTON, OHIO  45469
C
C
C     VARIABLE NAME               DESCRIPTION
C
C     N                           NUMBER OF ELEMENTS IN INPUT SKELETON
C                                 MATRIX.
C
C     MATRIX                      INPUT SKELETON MATRIX
C
C     INDEX                       INDEX SET OF SKELETON MATRIX
C
C     STAGES                      INPUT VECTOR DENOTING NUMBER OF
C                                 ELEMENTS ON EACH STAGE.  STAGES(I) =
C                                 THE NUMBER OF ELEMENTS ON STAGE #I
C
C     TTYOUT                      FORTRAN WRITE UNIT NUMBER FOR TELETYPE
C
C     SYS                         DIMENSION SIZES OF SYSTEM MATRICES
C
C---------
C
C     LIST                        SCRATCH VECTOR WHICH CONTAINS THE
C                                 INDEX NUMBER OF ELEMENTS RELATED TO
C                                 ELEMENT #I
C
C     COUNT                       NUMBER OF ELEMENTS IN "LIST".
C
C     ROW                         CURRENT ROW BEING PROCESSED
C
C     STAGE                       NUMBER OF THE CURRENT STAGE BEING
C                                 PROCESSED
C
C     NSTAGE                      THE TOTAL NUMBER OF STAGES
C
C     ISTART                      STARTING SEARCH INDEX FOR UPPER
C                                 TRIANGULAR MATRIX
C
C****NOTE****
C          THE DIMENSION OF "LIST" SHOULD BE EQUAL TO "SYS".
C
C
      IMPLICIT INTEGER*2 (A-Z)
      INTEGER N, INDEX
      DIMENSION INDEX(SYS), STAGES(SYS), LIST(128)
      LOGICAL*1 MATRIX(SYS,SYS)
C
C     BEGIN PROCESSING: INITIALIZE
C
      ROW   = 1
      STAGE = 0
      I     = 1
C
C     BEGIN FINDING ELEMENTS
C
    1 STAGE = STAGE + 1
      ROW   = ROW + STAGES(STAGE)
      WRITE(TTYOUT,100)  STAGE
C
C     SEE IF WE ARE PROCESSING LAST STAGE
C
    4 IF(STAGE .EQ. NSTAGE)  GOTO  2
C
C     PROCESS ELEMENTS FOR STAGE# STAGE
C
      COUNT  = 0
      ISTART = I + 1
      DO 3 J=ISTART,N
C
C     FIND ALL ELEMENTS THAT ELEMENT# I REACHES TO
C
      IF(.NOT. MATRIX(I,J))  GOTO  3
C
C     FOUND ONE, KEEP A RECORD OF IT
C
      COUNT       = COUNT + 1
      LIST(COUNT) = INDEX(J)
    3 CONTINUE
      IF(COUNT .EQ. 0)  GOTO  2
C
C     ALL DONE WITH ELEMENT #I, PRINT OUT
C
      WRITE(TTYOUT,101)  INDEX(I), (LIST(II), II=1,COUNT)
C
    5 I = I + 1
C
C     ALL DONE WITH STAGES PRINTOUT????
C
      IF(I .GT. N)  RETURN
C
C     ALL DONE WITH THIS STAGE????
C
      IF(I .EQ. ROW)  GOTO  1
C
C     CONTINUE PROCESSING THIS STAGE
C
      GOTO  4
C
C     PROCESS LAST STAGE
C
    2 WRITE(TTYOUT,102)  INDEX(I)
      GOTO  5
C
C     FORMATS
C
  100 FORMAT(1H-,10X,11HSTAGE NO.  ,I3/1H0)
  101 FORMAT(11X,I5,3H =>,18(7(I5,1H,)/20X))
  102 FORMAT(11X,I5)
```

```
      SUBROUTINE ELCONT(N,REA,INDEX,TTYIN,TTYOUT,SYS)
C
C     THIS SUBROUTINE PERFORMS THE ELEMENTARY CONTRACTION PROCESS
C
C     EDITED BY:  DAVID R. YINGLING, JR.
C                 ENGINEERING AND PUBLIC POLICY GROUP
C                 UNIVERSITY OF DAYTON
C                 DAYTON, OHIO  45469
C
      IMPLICIT INTEGER*2 (A-Z)
      INTEGER N, INDEX, INDH, NN
      LOGICAL FOUND1, FOUND2
      LOGICAL*1 REA(SYS,SYS), REAH(128,128)
      DIMENSION INDEX(SYS),   INDH(128),   NUMS(2), LO(128)
C
C     VARIABLE NAME               DESCRIPTION
C
C     N                           THE NUMBER OF ELEMENTS IN "REA"
C
C     REA                         ARGUMENT REACHABILITY MATRIX
C
C     INDEX                       INDEX SET OF "REA"
C
C     TTYIN                       FORTRAN READ UNIT NUMBER FOR TELETYPE
C
C     TTYOUT                      FORTRAN WRITE UNIT NUMBER FOR TELETYPE
C
C     SYS                         DIMENSION SIZES OF SYSTEM MATRICES
C
C---------
C
C     REAH                        SCRATCH REACHABILITY MATRIX
C
C     INDH                        INDEX SET OF "REAH"
C
C     LO                          SCRATCH VECTOR
C
C     IU                          POSITION OF ELEMENT U ON "REA"
C
C.    IV                          POSITION OF ELEMENT V ON "REA"
C
C     IIU                         POSITION OF ELEMENT U ON "REAH"
C
C     IIV                         POSITION OF ELEMENT V ON "REAH"
C
C     NEWNAM                      THE NEW INDEX VALUE FOR THE CONTRACTED
C                                 ELEMENTS
C
C     U                           ELEMENT # U TO BE CONTRACTED
C
C     V                           ELEMENT #V TO BE CONTRACTED
C
C
C     ****NOTE****
C      THE DIMENSIONS OF "INDH", "REAH",+"LO" SHOULD BE EQUAL TO "SYS"
C
C
C     ASK FOR TWO ELEMENTS
```

```
C
    1 WRITE(TTYOUT,100)
      CALL GETNUM(NUMS,2,TTYIN,TTYOUT)
      U = NUMS(1)
      V = NUMS(2)
      IF(U .EQ. 0 .OR. V .EQ. 0)  GOTO  2
C
C     CHECK TO SEE THAT U AND V ARE IN SYSTEM INDEX
C
      CALL FINDIT(N,U,V,IU,IV,INDEX,FOUND1,FOUND2,SYS)
      IF(FOUND1 .AND. FOUND2)  GOTO  3
C
C     U AND/OR V DOESN'T EXIST
C
      IF(.NOT. FOUND1)  WRITE(TTYOUT,101)  U
      IF(.NOT. FOUND2)  WRITE(TTYOUT,101)  V
      GOTO  1
C
C     DOES U REACH TO V ?
C
    3 NN = N
      CALL HIERCH(NN,REA,INDEX,REAH,INDH,NL,LO,SYS)
      CALL STAN(NN,REAH,INDH,NL,LO,SYS)
      CALL CONDE(NN,REAH,INDH,LO,TTYOUT,.FALSE.,SYS)
      CALL SKLTN(NN,REAH,SYS)
      CALL FINDIT(NN,U,V,IIU,IIV,INDH,FOUND1,FOUND2,SYS)
C
      IF(FOUND1 .AND. FOUND2 .AND. REAH(IIU,IIV))  GOTO  4
C
C     U IS NOT ADJACENT TO V, ISSUE ERROR MESSAGE
C
      WRITE(TTYOUT,102)  U, V
      GOTO  1
C
C     OK, GET NEW NAME
C
    4 WRITE(TTYOUT,103)
      CALL GETNUM(NUMS,1,TTYIN,TTYOUT)
      NEWNAM = NUMS(1)
      IF(NEWNAM .EQ. 0)  GOTO  2
C
C     CHECK TO SEE THAT NEWNAM IS NOT IN SYSTEM INDEX
C
      CALL FINDIT(N,NEWNAM,I,I,I,INDEX,FOUND1,FOUND2,SYS)
      IF(.NOT. FOUND1)  GOTO  5
C
C     NEWN AM IS IN THE INDEX SET, ISSUE ERROR MSG
C
      WRITE(TTYOUT,104)  NEWNAM
      GOTO  4
C
C     OK, NOW CHANGE MATRIX.
C
    5 CALL COMBIN(N,REA,INDEX,IU,IV,NEWNAM,SYS)
      CALL TRNCLS(N,REA,SYS)
      CALL IO(N,REA,INDEX,IO,.FALSE.,SYS)
      GOTO  1
C
```

```
    2 RETURN
C
C     FORMATS
C
  100 FORMAT(38H-TYPE TWO ELEMENTS TO BE CONTRACTED ? )
  101 FORMAT(12H-***ERROR***,I5,20H NOT IN SYSTEM INDEX)
  102 FORMAT(12H-***ERROR***,I5,19H IS NOT ADJACENT TU,I5)
  103 FORMAT(20H NEW INDEX NUMBER ? )
  104 FORMAT(12H-***ERROR***,I5,24H ALREADY IN SYSTEM INDEX)
      END
      SUBROUTINE POOL(N,REA,INDEX,TTYIN,TTYOUT,SYS)
C
C     THIS SUBROUTINE PERFORMS THE POOLING OPERATION
C
C     EDITED BY:  DAVID R. YINGLING, JR.
C                 ENGINEERING AND PUBLIC POLICY GROUP
C                 UNIVERSITY OF DAYTON
C                 DAYTON, OHIO  45469
C
C
C     VARIABLE NAME               DESCRIPTION
C
C     N                           NUMBER OF ELEMENTS IN "REA"
C
C     REA                         ARGUMENT REACHABILITY MATRIX
C
C     INDEX                       INDEX SET FOR "REA"
C
C     TTYIN                       FORTRAN READ UNIT NUMBER FOR TELETYPE
C
C     TTYOUT                      FORTRAN WRITE UNIT NUMBER FOR TELETYPE
C
C     SYS                         DIMENSION SIZES OF SYSTEM MATRICES
C
C-------------
C
C     END                         ENDING SUBSCRIPT OF LEVEL OR STAGE #I
C
C     IU                          POSITION OF ELEMENT U ON "REA"
C
C     IV                          POSITION OF ELEMENT V ON "REA"
C
C     INDH                        INDEX SET FOR "REAH"
C
C     LO                          SCRATCH VECTOR. LO(I) = NUMBER
C                                 OF ELEMENTS ON LEVEL # I (OR STAGE
C                                 # I IFF "STAGES" = .TRUE.)
C
C     NEWNAM                      THE NEW INDEX VALUE FOR THE
C                                 POOLED ELEMENTS
C
C     REAH                        SCRATCH REACHABILITY MATRIX
C
C     STAGES                      LOGICAL VARIABLE IF WHEN .TRUE.,
C                                 THE STAGES ARE BEING EXAMINED, WHEN
C                                 .FALSE., THE LEVELS ARE BEING
C                                 EXAMINED
C
C     START                       STARTING SUBSCRIPT OF LEVEL OR STAGE#I
C
C     U                           ELEMENT #1 TO BE POOLED
C
C     V                           ELEMENT #2 TO BE POOLED
C
      IMPLICIT INTEGER*2 (A-Z)
      INTEGER N, INDEX, INDH
      LOGICAL FOUND1, FOUND2, STAGES
      LOGICAL*1 REA(SYS,SYS), REAH(128,128)
      DIMENSION INDEX(SYS), INDH(128), LO(128), NUMS(2)
C
C     ****NOTE****
C      THE DIMENSIONS OF "REAH", "INDH", +"LO" SHOULD BE EQUAL TO "SYS".
C
C
C     ASK USER FOR ELEMENTS TO BE POOLED
C
    1 WRITE(TTYOUT,100)
      CALL GETNUM(NUMS,2,TTYIN,TTYOUT)
      U = NUMS(1)
      V = NUMS(2)
      IF(U .EQ. 0 .OR. V .EQ. 0)  GOTO  2
C
C     SEE IF U AND V EXIST IN SYSTEM INDEX
C
      CALL FINDIT(N,U,V,IU,IV,INDEX,FOUND1,FOUND2,SYS)
      IF(FOUND1 .AND. FOUND2)  GOTO  12
C
C     U AND/OR V DOESN'T EXIST
C
      IF(.NOT. FOUND1)  WRITE(TTYOUT,101)  U
      IF(.NOT. FOUND2)  WRITE(TTYOUT,101)  V
      GOTO  1
C
C     CHECK FOR U AND V ON SAME LEVEL
C
   12 STAGES = .FALSE.
      CALL HIERCH(N,REA,INDEX,REAH,INDH,NL,LO,SYS)
C
C     USE "LO" TO DETERMINE IF U AND V ARE ON SAME LEVEL OR STAGE.
C
   11 FOUND2 = .FALSE.
      FOUND1 = .FALSE.
      START  = 1
C
      DO 5 I=1,NL
      END = START + LO(I) - 1
C
      DO 6 J=START,END
      IF(INDH(J) .EQ. U)  FOUND1 = .TRUE.
      IF(INDH(J) .EQ. V)  FOUND2 = .TRUE.
    6 CONTINUE
C
      IF(FOUND1 .AND. FOUND2)  GOTO  3
C
      IF(FOUND1 .OR. FOUND2)  GOTO  7
C
      START = END + 1
    5 CONTINUE
C
C     NOT ON SAME LEVEL, SEE IF ON SAME STAGE
C
    7 IF(STAGES)  GOTO  13
C
C     TRANSPOSE MODEL MATRIX IN ORDER TO FOOL "HIERCH" SUBROUTINE
C
      DO 8 I=1,N
      DO 8 J=1,N
      REAH(I,J) = REA(J,I)
    8 CONTINUE
C
      DO 9 I=1,N
      DO 9 J=1,N
      REA(I,J) = REAH(I,J)
    9 CONTINUE
C
      CALL HIERCH(N,REA,INDEX,REAH,INDH,NL,LO,SYS)
C
C     TRANSPOSE MATRIX BACK SO IT IS NOT SCRAMBLED
C
      DO 10 I=1,N
      DO 10 J=1,N
      REA(I,J) = REAH(J,I)
   10 CONTINUE
      STAGES = .TRUE.
      GOTO  11
C
C     POOLING ERROR MESSAGE
C
   13 WRITE(TTYOUT,104)  U, V
      GOTO  1
C
C     GET PROPER "POSITIONS" FOR ELEMENTS U AND V ON "REA".
C
    3 CALL FINDIT(N,U,V,IU,IV,INDEX,FOUND1,FOUND2,SYS)
C
C     ASK FOR NEW NAME
C
      WRITE(TTYOUT,102)
      CALL GETNUM(NUMS,1,TTYIN,TTYOUT)
      NEWNAM = NUMS(1)
      IF(NEWNAM .EQ. 0)  GOTO  2
C
C     MAKE SURE NEWNAM DOESN'T EXIST IN SYSTEM INDEX
C
      CALL FINDIT(N,NEWNAM,J,J,J,INDEX,FOUND1,FOUND2,SYS)
      IF(.NOT. FOUND1)  GOTO  4
C
C     NEWNAM IS IN THE INDEX SET, ISSUE ERROR MESSAGE
C
      WRITE(TTYOUT,103) NEWNAM
      GOTO  3
C
C     OK, CHANGE THE MATRIX
C
    4 CALL COMBIN(N,REA,INDEX,IU,IV,NEWNAM,SYS)
      CALL TRNCLS(N,REA,SYS)
      GOTO  1
C
    2 RETURN
C
C     FORMATS
C
  100 FORMAT(34H-TYPE TWO ELEMENTS TO BE POOLED ? )
  101 FORMAT(12H-***ERROR***,I5,20H NOT IN SYSTEM INDEX)
```

```
      102 FORMAT(20H NEW INDEX NUMBER ? )
      103 FORMAT(12H-***ERROR***,I5,24H ALREADY IN SYSTEM INDEX)
      104 FORMAT(12H-***ERROR***,I5,4H AND,I5,35H ARE NOT ON THE SAME LEVEL
         +OR STAGE)
          END
      SUBROUTINE COMBIN(N,REA,INDEX,IU,IV,NEWNAM,SYS)
C
C     THIS SUBROUTINE COMBINES IU AND IV IN REA
C
C     EDITED BY:  DAVID R. YINGLING, JR.
C                 ENGINEERING AND PUBLIC POLICY GROUP
C                 UNIVERSITY OF DAYTON
C                 DAYTON, OHIO  45469
C                 513-228-2238
C
C
C     VARIABLE NAME               DESCRIPTION
C
C     N                           NUMBER OF ELEMENTS IN "REA"
C
C     REA                         ARGUMENT REACHABILITY MATRIX
C
C     INDEX                       INDEX SET FOR "REA"
C
C     IU                          SUBSCRIPT #1 TO BE COMBINED
C
C     IV                          SUBSCRIPT #2 TO BE COMBINED
C
C     NEWNAM                      THE INDEX VALUE FOR THE COMBINED
C                                 ELEMENTS
C
C     SYS                         DIMENSION SIZES OF SYSTEM MATRICES
C
      IMPLICIT INTEGER*2 (A-Z)
      INTEGER N, INDEX
      LOGICAL*1 REA(SYS,SYS)
      DIMENSION INDEX(SYS)
C
C     REPLACE ROW V WITH THE BOOLEAN SUM OF U AND V
C
      DO 1 J=1,N
      REA(IV,J) = REA(IU,J) .OR. REA(IV,J)
    1 CONTINUE
C
C     REPLACE COL V WITH THE BOOLEAN SUM OF U AND V
C
      DO 2 I=1,N
      REA(I,IV) = REA(I,IV) .OR. REA(I,IU)
    2 CONTINUE
C
C     REPLACE V'S INDEX WITH NEWNAM
C
      INDEX(IV) = NEWNAM
C
C     ERASE ROW, COL, AND INDEX FOR U
C
      CALL ELIM(N,REA,INDEX,IU,SYS)
      RETURN
      END
      SUBROUTINE TRNCLS(N,MATRIX,SYS)
C
C     THIS SUBROUTINE WILL TRANSITIVLY CLOSE THE INPUT MATRIX
C
C     WRITTEN BY: RAYMOND L. FITZ, S.M.
C      EDITED BY: DAVID R. YINGLING, JR.
C                 ENGINEERING AND PUBLIC POLICY GROUP
C                 UNIVERSITY OF DAYTON
C                 DAYTON, OHIO  45469
C
C     VARIABLE NAME                  DESCRIPTION
C
C     N                              THE NUMBER OF ELEMENTS IN "MATRIX"
C
C     MATRIX                         INPUT ADJACENCY MATRIX/OUTPUT
C                                    REACHABILITY MATRIX
C
C     SYS                            DIMENSION SIZES OF SYSTEM MATRICES
C
C------------
C
C     NONES                          NUMBER OF ONES IN THE REACHABILITY
C                                    SET OF ELEMENT #I
C
C     LAST                           NUMBER OF ONES IN THE REACHABILITY
C                                    SET OF ELEMENT #I FROM LAST
C                                    COMPUTATION
C
C     RECORD                         VECTOR USED TO KEEP ACCOUNT
C                                    OF ONES IN THE REACHABLILITY SET
C                                    OF ELEMENT #I
C
      IMPLICIT INTEGER*2 (A-Z)
      INTEGER N
      LOGICAL*1  MATRIX(SYS,SYS)
      DIMENSION  RECORD(128)
C
C     ****NOTE****
C      THE DIMENSION OF "RECORD" SHOULD BE EQUAL TO "SYS".
C
C     INITIALIZE PROCEDURE
C
      DO 1 I=1,N
C
C     PROCESS ELEMENT #I
C
      NONES = 0
      LAST  = 0
C
C     FIND ALL ONES IN REACHABILITY SET OF ELEMENT #I
C
    2 DO 3 K=1,N
      IF(.NOT. MATRIX(I,K))  GOTO  3
C
C     FOUND A ONE, KEEP A RECORD OF IT
C
      NONES         = NONES + 1
      RECORD(NONES) = K
```

```
C
    3 CONTINUE
C
C     CHECK TO SEE IF ANY NEW ONES WERE ADDED FROM LAST
C     TIME THROUGH
C
      IF(NONES .EQ. LAST)  GOTO  1
C
C     NO, COMPUTE NEW ELEMENTS IN REACHABILITY SET OF
C     ELEMENT #I BY TRANSITIVITY
C
      LAST = NONES
C
C     THE NEXT GROUP OF CODE PERFORMS THE PROCESS:
C     IF MATRIX(I,K)=1 .AND. MATRIX(K,J)=1,
C     THEN MATRIX(I,J)=1.
C
      DO 4 L=1,NONES
      K = RECORD(L)
      DO 4 J=1,N
      MATRIX(I,J) = MATRIX(I,J) .OR. MATRIX(K,J)
    4 CONTINUE
C
C     KEEP GOING UNTIL ALL REACHABILITY SETS HAVE BEEN EXAMINED
C
      NONES = 0
      GOTO  2
C
    1 CONTINUE
      RETURN
      END
      SUBROUTINE ADEDGE(N,REA,INDEX,TTYIN,TTYOUT,SYS)
C
C     THIS SUBROUTINE ADDS AN EDGE ON THE MINIMUM EDGE DIGRAPH
C
C     WRITTEN BY:  DAVID R. YINGLING, JR.
C                  ENGINEERING AND PUBLIC POLICY GROUP
C                  UNIVERSITY OF DAYTON
C                  DAYTON, OHIO  45469
C                  513-228-2238
C
C     VARIABLE NAME              DESCRIPTION
C
C     N                          NUMBER OF ELEMENTS IN "REA"
C
C     REA                        ARGUMENT REACHABILITY MATRIX
C
C     INDEX                      INDEX SET FOR "REA"
C
C     TTYIN                      FORTRAN READ UNIT NUMBER FOR TELETYPE
C
C     TTYOUT                     FORTRAN WRITE UNIT NUMBER FOR TELETYPE
C
C     SYS                        DIMENSION SIZES OF SYSTEM MATRICES
C
C-------------
C
C     N1                         INDEX VALUE OF ORGINATING EDGE ELEMENT
C
C     N2                         INDEX VALUE OF DESTINATION EDGE
C                                ELEMENT
C
C     II                         SUBSCRIPT OF "N1" ON "REA"
C
C     JJ                         SUBSCRIPT OF "N2" ON "REA"
C
      IMPLICIT INTEGER*2 (A-Z)
      INTEGER N, INDEX
      LOGICAL FOUND1, FOUND2
      LOGICAL*1 REA(SYS,SYS)
      DIMENSION INDEX(SYS), NUMS(2)
C
C     ASK USER FOR A REACHES TO B
C
      WRITE(TTYOUT,100)
    1 CALL GETNUM(NUMS,2,TTYIN,TTYOUT)
      N1 = NUMS(1)
      N2 = NUMS(2)
      IF(N1 .EQ. 0 .OR. N2 .EQ. 0)  GOTO  2
C
C     CHECK TO SEE IF N1 AND N2 ARE IN SYSTEM INDEX
C
      CALL FINDIT(N,N1,N2,II,JJ,INDEX,FOUND1,FOUND2,SYS)
      IF(FOUND1 .AND. FOUND2)  GOTO  3
C
C     N1 AND/OR N2 NOT IN SYSTEM INDEX
C
      IF(.NOT. FOUND1)  WRITE(TTYOUT,101)  N1
      IF(.NOT. FOUND2)  WRITE(TTYOUT,101)  N2
      GOTO  2
C
C     PUT IN EDGE
C
    3 REA(II,JJ) = .TRUE.
      GOTO  1
C
C     PERFORM TRANSITIVE CLOSURE
C
    2 CALL TRNCLS(N,REA,SYS)
      RETURN
C
C     FORMATS
C
  100 FORMAT(22H-TYPE <A REACHES TO B>)
  101 FORMAT(12H-***ERROR***,I5,20H NOT IN SYSTEM INDEX)
      END
      SUBROUTINE EREDGE(N,REA,INDEX,TTYIN,TTYOUT,SYS)
C
C     THIS SUBROUTINE ERASES AND EDGE ON THE MINIMUM EDGE DIGRAPH
C
C     WRITTEN BY:  DAVID R. YINGLING, JR.
C                  DECISIONS SYSTEMS LAB
C                  ENGINEERING AND PUBLIC POLICY GROUP
C                  UNIVERSITY OF DAYTON
C                  DAYTON, OHIO  45469
C
C
C
C     N                            NUMBER OF ELEMENTS IN "REA"
C
C     REA                          ARGUMENT REACHABILITY MATRIX
C
C     INDEX                        INDEX SET FOR "REA"
C
C     TTYIN                        FORTRAN READ UNIT NUMBER FOR TELETYPE
C
C     TTYOUT                       FORTRAN WRITE UNIT NUMBER FOR TELETYPE
C
C     SYS                          DIMENSION SIZES OF SYSTEM MATRICES
C
C---------------
C
C     N1                           INDEX VALUE OF ORGINATION OF EDGE TO
C                                  BE ERASED
C
C     N2                           INDEX VALUE OF DESTINATION OF EDGE
C                                  TO BE ERASED
C
      IMPLICIT INTEGER*2 (A-Z)
      INTEGER N, INDEX, INDH, NN
      LOGICAL FOUND1, FOUND2
      LOGICAL*1 REA(SYS,SYS), REAH(128,128)
      DIMENSION INDEX(SYS),   INDH(128),    LO(128),  NUMS(2)
C
C     ****NOTE****
C      THE DIMENSIONS OF "LO", "INDH", +"REAH" SHOULD BE EQUAL TO "SYS".
C
C
C     ASK USER FOR A REACHES TO B TO BE ERASED
C
      WRITE(TTYOUT,100)
    1 CALL GETNUM(NUMS,2,TTYIN,TTYOUT)
      N1 = NUMS(1)
      N2 = NUMS(2)
      IF(N1 .EQ. 0 .OR. N2 .EQ. 0)  GOTO  2
C
C     CHECK TO SEE IF N1 AND N2 ARE IN SYSTEM INDEX
C
      CALL FINDIT(N,N1,N2,II,JJ,INDEX,FOUND1,FOUND2,SYS)
      IF(FOUND1 .AND. FOUND2)  GOTO  3
C
C     N1 AND/OR N2 NOT IN SYSTEM INDEX, ISSUE ERROR MSG
C
      IF(.NOT. FOUND1)  WRITE(TTYOUT,101)  N1
      IF(.NOT. FOUND2)  WRITE(TTYOUT,101)  N2
    2 RETURN
C
C     OK, NOW CHECK FOR CYCLES
C
C     VARIABLE NAME                   DESCRIPTION
    3 IF(.NOT. (REA(II,JJ) .AND. REA(JJ,II)))  GOTO  4
C
C     WHUOPS, N1 AND N2 ARE IN A CYCLE, ISSUE ERROR MSG
C
      WRITE(TTYOUT,102)  N1, N2, N1
      GOTO  2
C
C     OK, CALCULATE SKELETON MATRIX
C
    4 NN = N
      CALL HIERCH(NN,REA,INDEX,REAH,INDH,NL,LO,SYS)
      CALL STAN(NN,REAH,INDH,NL,LO,SYS)
      CALL CONDE(NN,REAH,INDH,LO,TTYOUT,.FALSE.,SYS)
      CALL SKLTN(NN,REAH,SYS)
C
C     CHECK TO SEE IF N1 AND N2 ARE ON MINIMUM EDGE DIGRAPH
C
      CALL FINDIT(NN,N1,N2,II,JJ,INDH,FOUND1,FOUND2,SYS)
      IF(FOUND1 .AND. FOUND2)  GOTO  5
C
C     N1 AND/OR N2 WAS NOT ON MINIMUM EDGE DIGRAPH
C
      IF(.NOT. FOUND1)  WRITE(TTYOUT,103)  N1
      IF(.NOT. FOUND1)  WRITE(TTYOUT,103)  N2
      GOTO  2
C
C     CHECK TO SEE IF THE EDGE FROM N1 TO N2 EXISTS ON MINIMUM
C     EDGE DIGRAPH
C
    5 IF(REAH(II,JJ))  GOTO  6
C
C     THE EDGE FROM N1 TO N2 WAS NOT ON MINIMUM EDGE DIGRAPH
C     ISSUE ERROR MESSAGE
C
      WRITE(TTYOUT,104)  N1, N2
      GOTO  2
C
C     OK, N1 REACHES TO N2 ON MINIMUM EDGE DIGRAPH
C
C     ELIMINATE REACHABILITY BY DISCONNECTING ANTECEEDENT SET
C     OF N1 FROM THE REACHABILITY SET OF N2
C
    6 CALL FINDIT(N,N1,N2,II,JJ,INDEX,FOUND1,FOUND2,SYS)
C
C     REMOVE EDGE FROM L TO K IF:
C                                 1. L IS A MEMBER OF THE ANTECEEDENT SET
C                                    OF N1
C
C                                               AND
C
C                                 2. K IS A MEMBER OF THE REACHABILITY
C                                    SET OF N2
```

```
C
      DO 7 L=1,N
      IF(.NOT. REA(L,II))  GOTO  7
      DO 8 K=1,N
      IF(.NOT. REA(JJ,K))  GOTO  8
      REA(L,K) = .FALSE.
    8 CONTINUE
    7 CONTINUE
C
C     CALL TRANSITIVE CLOSURE
C
      CALL TRNCLS(N,REA,SYS)
      CALL IO(N,REA,INDEX,10,.FALSE.,SYS)
      GOTO  1
C
C     FORMATS
C
  100 FORMAT(35H-TYPE <A REACHES TO B> TO BE ERASED)
  101 FORMAT(12H-***ERROR***,I5,20H NOT IN SYSTEM INDEX)
  102 FORMAT(12H-***ERROR***,I5,4H AND,I5,16H ARE IN A CYCLE./7H "ELIM",
     +I5,36H AND RE-ENTFR USING THE "BO" COMMAND)
  103 FORMAT(12H-***ERROR***,I5,31H IS NOT ON MINIMUM EDGE DIGRAPH)
  104 FORMAT(26H-***ERROR*** THE EDGE FROM,I5,3H TO,I5,9H DOES NOT/30H E
     +XIST ON MINIMUM EDGE DIGRAPH)
      END
      SUBROUTINE ERASE(N,MATRIX,INDEX,TTYIN,TTYOUT,SYS)
C
C     THIS SUBROUTINE ERASES AN ELEMENT FROM "MATRIX".
C
C     WRITTEN BY: DAVID R. YINGLING, JR.
C                 ENGINEERING AND PUBLIC POLICY GROUP
C                 UNIVERSITY OF DAYTON
C                 DAYTON, OHIO  45469
C
C     VARIABLE NAME                   DESCRIPTION
C
C     N                               THE NUMBER OF ELEMENTS IN "MATRIX"
C
C     MATRIX                          THE MATRIX TO ERASE THE ELEMENT FROM
C
C     INDEX                           THE INDEX SET OF "MATRIX".
C
C     TTYIN                           FORTRAN READ UNIT NUMBER FOR TELETYPE
C
C     TTYOUT                          FORTRAN WRITE UNIT NUMBER FOR TELETYPE
C
C     SYS                             DIMENSION SIZES OF SYSTEM MATRICES
C
C
      IMPLICIT INTEGER*2 (A-Z)
      INTEGER N, INDEX
      LOGICAL*1 MATRIX(SYS,SYS)
      LOGICAL FOUND1, FOUND2
      DIMENSION INDEX(SYS), NUMS(1)
C
      IF(N .LE. 0)  GOTO  4
    1 WRITE(TTYOUT,100)
    2 CALL GETNUM(NUMS,1,TTYIN,TTYOUT)
      I = NUMS(1)
      IF(I .EQ. 0)  RETURN
C
C     CHECK TO SEE IF "I" IS A MEMBER OF THE INDEX SET
C
      CALL FINDIT(N,I,J,II,J,INDEX,FOUND1,FOUND2,SYS)
      IF(.NOT. FOUND1)  GOTO  3
C
C     NUMBER WAS VALID ELEMENT NUMBER, ERASE FROM MATRIX
C
      CALL ELIM(N,MATRIX,INDEX,II,SYS)
      IF(N .LE. 0)  GOTO  4
C
C     GO GET ANOTHER
C
      GOTO  2
C
C     ERROR PRINTOUT
C
    3 WRITE(TTYOUT,101)   I
      GOTO  1
C
C     USER HAS DELETED ALL ELEMENTS
C
    4 WRITE(TTYOUT,102)
```

```
      RETURN
C
C     FORMATS
C
  100 FORMAT(36H-TYPE ELEMENT NUMBERS TO BE ERASED ?)
  101 FORMAT(12H-***ERROR***,I5,20H NOT IN SYSTEM INDEX)
  102 FORMAT(53H-***NOTE*** THERE ARE NO ELEMENTS IN THE MODEL MATRIX/
     +26H "ELIM" COMMAND TERMINATED)
      END
      SUBROUTINE ADDEL(N,MATRIX,INDEX,TTYIN,TTYOUT,SYS)
C
C     THIS SUBROUTINE ADDS ELEMENTS TO THE MATRIX "MATRIX"
C     AND PUTS A "ONE" ON THE MAIN DIAGONAL.
C
C     WRITTEN BY: DAVID R. YINGLING, JR.
C                 ENGINEERING AND PUBLIC POLICY GROUP
C                 UNIVERSITY OF DAYTON
C                 DAYTON, OHIO  45469
C
C     VARIABLE NAME                     DESCRIPTION
C
C     N                                 THE NUMBER OF ELEMENTS IN "MATRIX"
C
C     MATRIX                            THE MATRIX TO ADD ELEMENTS TO
C
C     INDEX                             THE INDEX SET OF "MATRIX".
C
C     TTYIN                             FORTRAN READ UNIT NUMBER FOR TELETYPE
C
C     TTYOUT                            FORTRAN WRITE UNIT NUMBER FOR TELETYPE
C
C     SYS                               DIMENSION SIZES OF SYSTEM MATRICES
C
C
      IMPLICIT INTEGER*2 (A-Z)
      INTEGER N, INDEX
      LOGICAL*1  MATRIX(SYS,SYS)
      LOGICAL FOUND1, FOUND2
      DIMENSION INDEX(SYS), NUMS(1)
C
      IF(N .GE. SYS)  GOTO  5
    1 WRITE(TTYOUT,100)
    2 CALL GETNUM(NUMS,1,TTYIN,TTYOUT)
      I = NUMS(1)
      IF(I .EQ. 0)  RETURN
C
C     MAKE SURE THAT WE DON'T ALREAD HAVE AN ELEMENT # I
C
      CALL FINDIT(N,I,J,J,J,INDEX,FOUND1,FOUND2,SYS)
      IF(FOUND1)  GOTO  3
C
C     ALL OK, ADD ELEMENT AND PUT A ONE ON MAIN DIAGONAL
C
      N             = N + 1
      INDEX(N)      = I
C
C     ZERO OUT ROW AND COL OF NEW ELEMENT
C
      DO 4 J=1,N
      MATRIX(J,N) = .FALSE.
      MATRIX(N,J) = .FALSE.
    4 CONTINUE
C
      MATRIX(N,N) = .TRUE.
C
C     GO GET ANOTHER ELEMENT
C
```

```
      IF(N .GE. SYS)  GOTO  5
      GOTO  2
C
C     ERROR PRINTOUT
C
    3 WRITE(TTYOUT,101)  I
      GOTO  1
C
C     USER HAS EXCEEDED SYSTEM MATRIX SIZE
C
    5 WRITE(TTYOUT,102)  SYS
      RETURN
C
C     FORMATS
C
  100 FORMAT(28H-TYPE ELEMENTS TO BE ADDED ?)
  101 FORMAT(12H-***ERROR***,I5,24H ALREADY IN SYSTEM INDEX)
  102 FORMAT(62H-***NOTE*** NUMBER OF ELEMENTS HAS REACHED COMPUTER'S LI
     +MIT OF,I5/25H "ADD" COMMAND TERMINATED)
      END
      PROGRAM CYCLE
C
C     ***************************************************************
C     *                          NOTICE                             *
C     *                                                             *
C     *  ALL RIGHTS RESERVED.  NO PART OF THIS PROGRAM MAY BE SOLD, *
C     *  REPRODUCED, STORED IN A RETRIEVAL SYSTEM, OR TRANSMITTED   *
C     *  IN ANY FORM OR BY ANY MEANS, ELECTRONIC, MECHANICAL,       *
C     *  PHOTOCOPYING, RECORDING, OR OTHERWISE, WITHOUT THE         *
C     *  PRIOR PERMISSION OF THE                                    *
C     *      UNIVERSITY OF DAYTON RESEARCH INSTITUTE.               *
C     *                                                             *
C     ***************************************************************
C
C
C     PROGRAM CYCLE (SYS04)
C
C     PROGRAM THAT INITIALIZES A WEIGHTED MATRIX
C     AND RESOLVES THE THRESHOLD WITH WITH GEODEDIC OUTPUT
C
C     WRITTEN BY:  DAVID R. YINGLING, JR.
C                  DECISION SYSTEMS LAB
C                  ENGINEERING AND PUBLIC POLICY GROUP
C                  UNIVERSITY OF DAYTON
C                  DAYTON, OHIO  45469
C                  513-229-2238
C
      IMPLICIT   INTEGER  (A-Z)
      COMMON /INFO/  QTYPE,  TTYIN,  TTYOUT
      COMMON /BLK1/  L1(160), L2(160)
C
      DIMENSION  IM(50,50),   INDEX(50),   NUMS(3), BULL(5704)
      LOGICAL  QTYPE
      DATA  CH/2HCH/,  AL/2HAL/,  DL/2HDL/,  FI/2HFI/
      DATA  PR/2HPR/,  RE/2HRE/,  TE/2HTE/,  HELP/2HHE/
      DATA  YES/1HY/,  NO/1HN/
C
C     INITIALIZATIONS
C
      TTYIN = 5
      TTYOUT= 2
      MWEIGH = 9
      QTYPE = .TRUE.
      PERMFL= 20
C
C     MAIN CONTROL SECTION
C
C
C     FULL TEXT ?
C
      WRITE(TTYOUT,2)
      READ(TTYIN,3)   ANSWER
      IF(ANSWER .EQ. YES)  QTYPE = .FALSE.
C
C     NEW SYSTEM ?
C
      WRITE(TTYOUT,4)
      READ(TTYIN,3)   ANSWER
      IF(ANSWER .EQ. NO)  GOTO  10
C
C     COME HERE WHEN NEW SYSTEM
C
   11 WRITE(TTYOUT,5)
      CALL GETNUM(NUMS,1)
      N = NUMS(1)
      IF(N .LE. 50)  GOTO  12
      WRITE(TTYOUT,7)
      GOTO  11
C
C     READ IN FROM PERMFL OLD SYSTEM
C
   10 REWIND 20
      READ(20)  N
      READ(20)  IM
      READ(20)  INDEX
      GOTO  15
C
C     ASK IF USER WANTS REGULAR INDEXING
C
   12 CONTINUE
      WRITE(TTYOUT,8)
      READ(TTYIN,3)  ANSWER
      IF(ANSWER .EQ. NO)  GOTO  13
C
C     PUT IN REGULAR INDEXES
C
      DO 14 I=1,N
      INDEX(I) = I
   14 CONTINUE
      GOTO  15
C
C     READ IRREGULAR INDEXES
C
   13 CONTINUE
      WRITE(TTYOUT,9)
      DO 16 I=1,N
      CALL GETNUM(NUMS,1)
      INDEX(I) = NUMS(1)
      IF(I .EQ. 1)   GOTO 16
      M = I - 1
      DO 46 J = 1,M
      IF(INDEX(J) .EQ. INDEX(I))   WRITE(TTYOUT,103) INDEX(I)
      IF(INDEX(J) .EQ. INDEX(I))   I = I - 1
   46 CONTINUE
   16 CONTINUE
C
   15 REWIND  20
      WRITE(20)  N
      WRITE(20)  IM
      WRITE(20)  INDEX
C
C     BEGIN FILLING THE ADJACENCY MATRIX
C     ASK USER FOR A COMMAND KEYWORD AND CHECK
C
      WRITE(TTYOUT,102)
      READ(TTYIN,100)  CMD
```

```
C
      IF(CMD .EQ. CH)    GOTO  17
      IF(CMD .EQ. AL)    GOTO  18
      IF(CMD .EQ. FI)    GOTO  19
      IF(CMD .EQ. DL)    GOTO  20
      IF(CMD .EQ. HELP)  GOTO  21
      IF(CMD .EQ. RE)    GOTO  22
      IF(CMD .EQ. TE)    GOTO  23
      IF(CMD .EQ. PR)    GOTO  45
C
      WRITE(TTYOUT,101)  CMD
      GOTO  15
C
C     *******************************************************************
C     *                         CHANGE WEIGHT ROUTINE                   *
C     *******************************************************************
   17 CALL CHANGE(N,IM,INDEX)
      GOTO  15
C     *******************************************************************
C     *                         ADD ELEMENT ROUTINE                     *
C     *******************************************************************
   18 CALL ADD(N,IM,INDEX)
      GOTO  15
C     *******************************************************************
C     *                         DELETE ELEMENT ROUTINE                  *
C     *******************************************************************
   20 CALL DELETE(N,IM,INDEX)
      GOTO 15
C     *******************************************************************
C     *                         FILL SYSTEM WITH WEIGHTS                *
C     *******************************************************************
   19 CALL FILL(N,IM,INDEX,R1,R2)
      GOTO  15
C     *******************************************************************
C     *                         PRINT SYSTEM OUT                        *
C     *******************************************************************
   45 CALL PRNTMT(N,IM,INDEX,THRESH,.FALSE.)
      GOTO  15
C     *******************************************************************
C     *                         RESOLVE SYSTEM                          *
C     *******************************************************************
   22 CALL RESOLV(N,IM,INDEX,MWEIGH)
      GOTO  15
C     *******************************************************************
C     *                         TERMINATION                             *
C     *******************************************************************
   23 CALL EXIT
C     *******************************************************************
C     *                         HELP !!!                                *
C     *******************************************************************
   21 WRITE(TTYOUT,104)
      GOTO  15
C
C     *******
C     FORMATS
C     *******
C
    2 FORMAT(1H0,35H FULL TEXT QUERIES DESIRED ? (Y/N) )
    3 FORMAT(A1)
    4 FORMAT(1H ,20H NEW SYSTEM ? (Y/N) )
    5 FORMAT(1H ,31H NUMBER OF ELEMENTS (50 MAX.) ?)
    7 FORMAT(1H ,27H ***TOO LARGE, TRY AGAIN***)
    8 FORMAT(1H ,46H REGULAR INDEXING OF ELEMENTS DESIRED ? (Y/N) )
    9 FORMAT(1H ,28H ENTER INDEXES ONE AT A TIME)
  100 FORMAT(A2)
  101 FORMAT(1H-,31H ***ERROR*** INVALID COMMAND==>,A2)
  102 FORMAT(1H-,32H TYPE CYCLE COMMAND (OR "HELP") )
  103 FORMAT(1H ,34H ELEMENT HAS ALREADY BEEN ENTERED>,I2)
  104 FORMAT(1H-,10X,18H***HELP MESSAGE***/1H0,30H AL - ADD AN ELEMENT T
     *O SYSTEM/1H ,35H DL - DELETE AN ELEMENT FROM SYSTEM/1H ,34H FI - F
     *ILL SYSTEM (ASSIGN WEIGHTS)/1H ,37H CH - CHANGE WEIGHT OF A RELATI
     *ONSHIP/1H ,20H RE - RESOLVE SYSTEM/1H ,22H PR - PRINT SYSTEM OUT/1
     *H ,23H TE - TERMINATE SESSION/1H ,27H HELP - REPRINTS ABOVE LIST)
      END
```

352

```
      SUBROUTINE PRNIMT(N,MAT,INDEX,THRESH,SELPNT)
C
C     THIS SUBROUTINE PRINTS ALL RELATIONSHIPS
C      >= TO THE THRESHOLD
C
C
C     WRITTEN BY:  DAVID R. YINGLING, JR.
C                  DECISION SYSTEMS LAB
C                  ENGINEERING AND PUBLIC POLICY GROUP
C                  UNIVERSITY OF DAYTON
C                  DAYTON, OHIO  45469
C                  513-229-2238
C
      IMPLICIT   INTEGER  (A-Z)
      COMMON /INFO/  QTYPE,  TTYIN,  TTYOUT
C
      INTEGER  INDEX(50),  MAT(50,50),  LIST(50,2)
      LOGICAL  QTYPE,  SELPNT
C
      IF(SELPNT)   WRITE(TTYOUT,1)
C
      DO 10 I=1,N
      CTR = 0
      DO 11 J=1,N
      IF(.NOT. SELPNT)   GOTO  13
      IF(MAT(I,J) .LT. THRESH)  GOTO  11
   13 IF(I .EQ. J)  GOTO  11
      CTR = CTR + 1
      LIST(CTR,1) = INDEX(J)
      LIST(CTR,2) = MAT(I,J)
   11 CONTINUE
C
      IF(CTR .EQ. 0)  GOTO  12
C
      WRITE(TTYOUT,2)  INDEX(I),(LIST(II,1),LIST(II,2),II = 1,CTR)
      GOTO  10
C
   12 WRITE(TTYOUT,3)  INDEX(I)
C
   10 CONTINUE
      RETURN
C
    2 FORMAT(1H ,I5,2H=>,8(7(I5,1H(,I1,2H),)/9X))
    3 FORMAT(I2,2H=>)
    1 FORMAT(1H-,15X,17H THRESHOLD MATRIX)
      END
      SUBROUTINE QUEST(LL1, LL2)
C
C     THIS SUBROUTINE PRESENTS THE QUESTIONS
C
C
C     WRITTEN BY:  DAVID R. YINGLING, JR.
C                  DECISION SYSTEMS LAB
C                  ENGINEERING AND PUBLIC POLICY GROUP
C                  UNIVERSITY OF DAYTON
C                  DAYTON, OHIO  45469
C                  513-229-2238
C
      IMPLICIT   INTEGER  (A-Z)
      COMMON /INFO/  QTYPE,  TTYIN,  TTYOUT
      COMMON /BLK1/  L1(160), L2(160)
C
      LOGICAL  QTYPE
C
      IF(QTYPE)  GOTO  15
      DEFINE FILE 8(260,160,U,UNUSED)
      I1 = LL1 + 4
      I2 = LL2 + 4
      READ(8'I1)  (L1(I),I=1,160)
      READ(8'I2)  (L2(I),I=1,160)
      I1 = 0
      I2 = 0
      DO 12 J=1,10
      IF(L1(J) .EQ. 0)  GOTO  12
      L = L1(J)
      I1 = I2 + 1
      I2 = I1 + L - 1
      WRITE(TTYOUT,2)  (L1(K + 10),K = I1,I2)
   12 CONTINUE
      WRITE(TTYOUT,3)
      I1 = 0
      I2 = 0
      DO 13 J = 1,10
      IF(L2(J) .EQ. 0)  GOTO  13
      L = L2(J)
      I1 = I2 + 1
      I2 = I1 + L - 1
      WRITE(TTYOUT,2)  (L2(K + 10),K = I1,I2)
   13 CONTINUE
      WRITE(TTYOUT,1)
   14 RETURN
   15 WRITE(TTYOUT,4)  LL1, LL2
      GOTO  14
C
    1 FORMAT(10H WEIGHT ? )
    2 FORMAT(1X,15A4)
    3 FORMAT(1H ,20H HAS BEEN RELATED TO)
    4 FORMAT(1X,I5,2H R,I4,9H WEIGHT ? )
      END
      SUBROUTINE ADD(N,MAT,INDEX)
C
C     THIS SUBROUTINE ADDS ELEMENTS TO THE ADJACENCY MATRIX
C
C     WRITTEN BY:  DAVID R. YINGLING, JR.
C                  DECISION SYSTEMS LAB
C                  ENGINEERING AND PUBLIC POLICY GROUP
C                  UNIVERSITY OF DAYTON
C                  DAYTON, OHIO  45469
C                  513-229-2238
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION NUMS(3), MAT(50,50), INDEX(50)
C
      COMMON /BLK1/  L1(160), L2(160)
      COMMON /INFO/  QTYPE,  TTYIN,  TTYOUT
      DATA NOO/1HN/, NO/2HNO/, Y/1HY/, YES/3HYES/
C
      LOGICAL FOUND1, FOUND2, QTYPE
C
C
C     ASK FOR ADDED ELEMENT NUMBER(S)
C
    2 WRITE(TTYOUT,106)
      CALL GETNUM(NUMS,1)
      N = N + 1
      INDEX(N) = NUMS(1)
      IF(INDEX(N) .LE. 9999)  GOTO  1
      WRITE(TTYOUT,100)   INDEX(N)
   12 N = N - 1
      GOTO  2
C
C     CHECK FOR A ZERO INPUT
C
    1 IF(INDEX(N) .GT. 0)  GOTO  3
      N = N - 1
      RETURN
C
C     CHECK FOR DUPLICATE ELEMENT
C
    3 K = N - 1
      CALL FINDIT(K,INDEX(N),J,J,J,INDEX,FOUND1,FOUND2)
      IF(.NOT. FOUND1)  GOTO  11
      WRITE(TTYOUT,101)   INDEX(N)
      GOTO  12
   11 CONTINUE
C
C     SEE IF USER WANTS TO BORDER
C
      WRITE(TTYOUT,102)
      READ(TTYIN,103)   ANSWER
      IF(ANSWER .EQ. Y)     GOTO  4
      IF(ANSWER .EQ. YES)   GOTO  4
      IF(ANSWER .EQ. NOO)   GOTO  2
      IF(ANSWER .EQ. NO)    GOTO  2
      WRITE(TTYOUT,104)   ANSWER
      GOTO  3
C
```

```
C     BORDER INPUT
C
    4 DO 5 I=1,K
    6 CALL QUEST(INDEX(I), INDEX(N))
      CALL GETNUM(NUMS,1)
      WEIGHT = NUMS(1)
      IF(WEIGHT .LE. 9)  GOTO  7
      WRITE(TTYOUT,105)   WEIGHT
      GOTO  6
    7 MAT(I,N) = WEIGHT
    5 CONTINUE
C
      DO 8 I=1,K
    9 CALL QUEST(INDEX(N), INDEX(I))
      CALL GETNUM(NUMS,1)
      WEIGHT = NUMS(1)
      IF(WEIGHT .LE. 9)  GOTO  10
      WRITE(TTYOUT,105)   WEIGHT
      GOTO  9
   10 MAT(N,I) = WEIGHT
    8 CONTINUE
      GOTO  2
C
C     *******
C     FORMATS
C     *******
C
  100 FORMAT(1H0,22H ***ELEMENT TOO LARGE>,I5,3H***)
  101 FORMAT(1H0,30H ***ELEMENT ALREADY IN SYSTEM>,I5,3H***)
  102 FORMAT(1H ,31H BORDER ON THIS ELEMENT ? (Y/N))
  103 FORMAT(A2)
  104 FORMAT(1H ,21H ***INVALID RESPONSE>,A2,3H***)
  105 FORMAT(1H ,28H *** WEIGHT VALUE TOO LARGE>,I5,3H***)
  106 FORMAT(1H ,29H ELEMENT NUMBER TO BE ADDED ?)
      END
      SUBROUTINE CHANGE(N,MAT,INDEX)
C
C     THIS SUBROUTINE WILL CALLOW THE USER TO CHANGE
C     A WEIGHT IN THE ADJACENCY MATRIX
C
C
C     WRITTEN BY:  DAVID R. YINGLING, JR.
C                  DECISION SYSTEMS LAB
C                  ENGINEERING AND PUBLIC FOLICY GROUP
C                  UNIVERSITY OF DAYTON
C                  DAYTON, OHIO  45469
C                  513-229-2238
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION MAT(50,50), INDEX(50), NUMS(3)
      COMMON /INFO/  QTYPE,  TTYIN,  TTYOUT
C
      LOGICAL FOUND1, FOUND2
C
C
    5 WRITE(TTYOUT,100)
      CALL GETNUM(NUMS,3)
      GOTO  6
    1 CALL GETNUM(NUMS,3)
C
C     CHECK FOR ZERO INPUT
C
    6 IF(NUMS(1) .GT. 0 .AND. NUMS(2) .GT. 0)  GOTO  8
      RETURN
C
C     CHECK FOR EXISTANCE OF ELEMENTS
C
    8 CALL FINDIT(N,NUMS(1),NUMS(2),X,Y,INDEX,FOUND1,FOUND2)
C
C     DID WE GET A VALID INPUT ?
C
      IF(FOUND1 .AND. FOUND2)  GOTO  3
      IF(Y .EQ. 0)  GOTO  4
      WRITE(TTYOUT,101)   NUMS(1)
      IF(Y .GT. 0)  GOTO  5
    4 WRITE(TTYOUT,101)   NUMS(2)
      GOTO  5
C
    3 IF(NUMS(3) .LE. 9)  GOTO  7
      WRITE(TTYOUT,102)   NUMS(3)
      GOTO  5
C
    7 MAT(X,Y) = NUMS(3)
      GOTO  1
C
C     *******
C     FORMATS
C     *******
C
  100 FORMAT(1H ,41H ENTER A REACHES TO B, WEIGHT (3 NUMBERS))
  101 FORMAT(1H ,28H ***ELEMENT DOES NOT EXIST>,I5,3H***)
  102 FORMAT(1H ,28H ***WEIGHT VALUE TOO LARGE>,I5,3H***)
      END
      SUBROUTINE DELETE(N,MAT,INDEX)
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION MAT(50,50), INDEX(50), NUMS(3)
      COMMON /INFO/  QTYPE,  TTYIN,  TTYOUT
C
C
      LOGICAL FOUND1, FOUND2
C
C
      WRITE(TTYOUT,101)
    1 CALL GETNUM(NUMS,1)
C
C     IS IT ZERO ?
C
      IF(NUMS(1) .EQ. 0)  RETURN
C
C     ELEMENT IN SET ?
C
      CALL FINDIT(N,NUMS(1),J,I,J,INDEX,FOUND1,FOUND2)
      IF(FOUND1)  GOTO  3
      WRITE(TTYOUT,100)   NUMS(1)
      GOTO  1
C
    3 IF(I .EQ. N)  GOTO  4
C
      N3 = N - 1
      DO 5 K1=1,N3
      K2 = K1 + 1
      DO 6 J=1,N
      MAT(K1,J) = MAT(K2,J)
    6 CONTINUE
      DO 7 J=1,N
      MAT(J,K1) = MAT(J,K2)
    7 CONTINUE
    5 CONTINUE
      DO 8 K1=1,N3
      K2 = K1 + 1
      INDEX(K1) = INDEX(K2)
    8 CONTINUE
C
    4 N = N - 1
      GOTO  1
C
C     *******
C     FORMATS
C     *******
C
  100 FORMAT(1H ,27H ***ELEMENT DOES NOT EXIST>,I5,3H***)
  101 FORMAT(1H ,30H ELEMENT NUMBER TO BE ERASED ?)
      END
      SUBROUTINE FILL(N,MAT,INDEX,R1,R2)
C
C     THIS SUBROUTINE ALLOWS THE USER TO FILL UP THE ADJACENCY MATRIX
C
C
C     WRITTEN BY:  DAVID R. YINGLING, JR.
C                  DECISION SYSTEMS LAB
C                  ENGINEERING AND PUBLIC POLICY GROUP
C                  UNIVERSITY OF DAYTON
C                  DAYTON, OHIO  45469
C                  513-229-2238
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION MAT(50,50), INDEX(50), NUMS(3)
      COMMON /INFO/  QTYPE,  TTYIN,  TTYOUT
      COMMON /BLK1/  L1(160), L2(160)
      LOGICAL QTYPE
C     SEE IF THIS IS A RESTART
C
      IF(R1 .GT. 0 .OR. R2 .GT. 0)  GOTO  1
      ROW = 1
      COL = 1
      R1  = 0
      R2  = 0
C
   10 DO 2 I=ROW,N
      DO 2 J=COL,N
      IF(I .EQ. J)  GOTO  2
    5 CALL QUEST(INDEX(I),INDEX(J))
      CALL GETNUM(NUMS,1)
      IF(NUMS(1) .EQ. 10)  GOTO  3
      IF(NUMS(1) .LE. 9)   GOTO  4
      WRITE(TTYOUT,101)   NUMS(1)
      GOTO  5
    4 MAT(I,J) = NUMS(1)
    2 CONTINUE
      GOTO  6
C
C     RESTART
C
    1 I = R1
      DO 7 J=R2,N
    8 CALL QUEST(INDEX(I),INDEX(J))
      CALL GETNUM(NUMS,1)
      IF(NUMS(1) .EQ. 10)  GOTO  3
      IF(NUMS(1) .LE. 9)   GOTO  9
      WRITE(TTYOUT,101)   NUMS(1)
      GOTO  8
    9 MAT(I,J) = NUMS(1)
    7 CONTINUE
      ROW = 1
      COL = R1 + 1
      R1  = 0
      R2  = 0
      GOTO  10
C
C     GENERATE RESTART PARAMS
C
```

```
    3 R1 = I
      R2 = J
    6 RETURN
C
C     *******
C     FORMATS
C     *******
C
  100 FORMAT(1H0,I5,2HR ,I5,10H, WEIGHT ?)
  101 FORMAT(1H ,28H ***WEIGHT VALUE TOO LARGE>,I5,3H***)
      END
      SUBROUTINE FINDIT(N,N1,N2,S1,S2,INDEX,FOUND1,FOUND2)
C
C     THIS SUBROUTINE FINDS ELEMENTS N1, N2 IN THE INDEX SFT
C
C     THE VALUES S1, S2 ARE THE POSITIONS OF N1 AND N2 IN THE
C     INDEX SET.
C
C     FOUND1 AND FOUND2 ARE LOGICAL VALUES AND ARE SET EQUAL
C     TO .TRUE. IF N1 OR N2 (RESPECTIVELY) ARE FOUND IN THE
C     INDEX SFT.
C
C     WRITTEN BY:  DAVID R. YINGLING, JR.
C                  DECISION SYSTEMS LAB
C                  ENGINEERING AND PUBLIC POLICY GROUP
C                  UNIVERSITY OF DAYTON
C                  DAYTON, OHIO  45469
C                  513-229-2238
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION INDEX(128)
      LOGICAL FOUND1, FOUND2
C
      FOUND1 = .FALSE.
      FOUND2 = .FALSE.
      S1     = 0
      S2     = 0
C
      DO 1 I=1,N
      IF(N1 .EQ. INDEX(I))  S1 = I
      IF(N2 .EQ. INDEX(I))  S2 = I
    1 CONTINUE
C
      IF(S1 .GT. 0)  FOUND1 = .TRUE.
      IF(S2 .GT. 0)  FOUND2 = .TRUE.
      RETURN
      END
      SUBROUTINE GETNUM(ARRAY,N)
C
C     THIS SUBROUTINE WILL READ "N" UNSIGNED INTEGERS FROM
C     THE TERMINAL TYPED IN A FREE FORMAT AND STORE THEM IN
C     "ARRAY"
C
C     WRITTEN BY:  DAVID R. YINGLING, JR.
C                  DECISION SYSTEMS LAB
C                  ENGINEERING AND PUBLIC POLICY GROUP
C                  UNIVERSITY OF DAYTON
C                  DAYTON, OHIO  45469
C                  513-229-2238
C
      IMPLICIT INTEGER (A-Z)
      COMMON /INFO/  QTYPE,  TTYIN,  TTYOUT
      DIMENSION ARRAY(1), BUFFER(80), NUMS(10)
      DATA NUMS/1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9/
      DATA BLANK/1H /, COMMA/1H,/
C
      DO 8 I=1,N
      ARRAY(I) = 0
    8 CONTINUE
C
    1 READ(TTYIN,200)  BUFFER
C
      L         = N
      POWER     = 0
C
      DO 2 I=1,80
      K = 81 - I
      IF(BUFFER(K) .EQ. BLANK .OR. BUFFER(K) .EQ. COMMA)  GOTO  3
C
C     FOUND A CHARACTER, SEE IF IT'S A VALID NUMERIC
C
      DO 4 J=1,10
      II = J - 1
      IF(BUFFER(K) .NE. NUMS(J))  GOTO  4
C
C     ITS A NUMBER, ADD IT TO PRESENT SUM
C
      ARRAY(L) = ARRAY(L) + (II * (10**POWER))
      POWER    = POWER + 1
      GOTO  2
    4 CONTINUE
C
C     COME HERE IF CHARACTER FOUND WAS NOT NUMERIC
C
      WRITE(TTYOUT,100)
      GOTO  1
C
C     FOUND A DELIMITER, SEE IF END OF A NUMBER
C
    3 IF(ARRAY(L) .EQ. 0)  GOTO  2
      L = L - 1
      IF(L .EQ. 0)  GOTO  5
      ARRAY(L) = 0
      POWER    = 0
    2 CONTINUE
```

```
C
C     MAKE SURE NUMBER(S) IS/ARE LESS THAN 99999 SO
C     WE DON'T EXCEED I5 FORMATS
C
    5 DO 6 I=1,N
      IF(ARRAY(I) .GT. 99999)  GOTO  7
    6 CONTINUE
      RETURN
C
C     ERROR MESSAGE
C
    7 WRITE(TTYOUT,101)
      GOTO  1
C
C     FORMATS
C
  100 FORMAT(37H-***ERROR*** INPUT NOT NUMERIC--RETRY)
  101 FORMAT(39H-***ERROR*** NUMBER(S) TOO LARGE--RETRY)
  200 FORMAT(80A1)
      END
      SUBROUTINE RESOLV(N,IN,LIST,THRESH)
C
C     THIS SUBROUTINE RESOLVES THE WEIGHTED MATRIX
C
      COMMON /INFO/  QTYPE,  TTYIN,  TTYOUT
C
      IMPLICIT   INTEGER  (A-Z)
      INTEGER  IN(50,50),  LIST(50)
      LOGICAL  ADJ(50,50),  QTYPE,  RM(50,50)
C
      T = THRESH
C
C     CONSTRUCT BINARY ADJACENCY MATRIX
C
   15 DO 10 I=1,N
      DO 10 J=1,N
      ADJ(I,J) = .FALSE.
      IF(I .EQ. J)  GOTO  11
      IF(IN(I,J) .LT. T)  GOTO  10
   11 ADJ(I,J) = .TRUE.
   10 CONTINUE
C
C     CHECK IF CYCLE IS RESOLVED AT THIS THRESHOLD
C
      CALL TRNCLS(ADJ,RM,N)
C
C     CHECK TO SEE IF REACHABILITY MAT IS
C     ALL ONES, IF SO, CYCLE IS RESOLVED
C     IF NOT, T = T - 1  AND CONSTRUCT NEW ADJ MATRIX
C
      DO 12 I=1,N
      DO 12 J=1,N
      IF(.NOT. RM(I,J))  GOTO  13
   12 CONTINUE
C
C     TELL USER CYCLE IS RESOLVED
C
      WRITE(TTYOUT,1)  T
C
C     PRINT UNIVERSAL MATRIX
C
      CALL PRNTMT(N,IN,LIST,T,.TRUE.)
C
C     PRINT GEODEDIC OUTPUT
C
      CALL GEOD(ADJ,N,LIST)
      RETURN
C
C     NO GOOD, TRY AGAIN
C
   13 T = T - 1
      GOTO  15
C
    1 FORMAT(1H-,' ***CYCLE RESOLVED***',/,' THRESHOLD==>',2X,I2)
      END
      SUBROUTINE TRNCLS(A,B,N)
C
C     THIS SUBROUTINE TAKES THE TRANSITIVE CLOSURE
C     OF MAT A AND PUTS THE ANSWER INTO MAT B.
C
      IMPLICIT   INTEGER  (A-Z)
      INTEGER  C(50)
      LOGICAL  A(50,50),  B(50,50)
C
      DO 1 I=1,N
      DO 1 J=1,N
    1 B(I,J) = A(I,J)
C
      I = 0
    2 N1 = 0
      NT = 0
      I = I + 1
      IF(I .GT. N)  RETURN
    5 DO 3 K=1,N
      IF(.NOT. B(I,K))  GOTO  3
      NT = NT + 1
      C(NT) = K
    3 CONTINUE
C
C
      IF(NT .EQ. N1)  GOTO  2
C
      N1 = NT
      DO 4 L=1,N1
      K = C(L)
      DO 4 J=1,N
    4 IF(B(K,J))  B(I,J) = .TRUE.
C
      NT = 0
      GOTO  5
      END
      SUBROUTINE GEOD(A,N,INDEX)
C
C     THIS SUBROUTINE PRINTS OUT GEODEDIC CYCLES
C
      COMMON /INFO/  QTYPE,  TTYIN,  TTYOUT
C
      IMPLICIT   INTEGER  (A-Z)
      INTEGER  B(50,50),  C(50,50),  G(50,50),  LIM(50)
      INTEGER  INDEX(50),  PRINT(50)
      LOGICAL  QTYPE,  A(50,50)
C
      DATA  G/2500*0/,  B/2500*0/
C
C     FORM  A - I
C
C
      DO 10 I=1,N
      DO 10 J=1,N
      IF(.NOT. A(J,I))  GOTO  10
      B(J,I) = 1
      G(J,I) = 1
      IF(J .EQ. I)  G(J,I) = 0
   10 CONTINUE
C
C     FORM  G
C
      NBN = 1
   11 DO 12 I=1,N
      DO 12 J=1,N
      C(J,I) = 0
      DO 13 K=1,N
      MA = 0
      IF(A(J,K))  MA = 1
   13 C(J,I) = MA * B(K,I) + C(J,I)
      IF(C(J,I) .NE. 0)  C(J,I) = 1
   12 CONTINUE
C
C     COMPUTE TO THE NTH
C
      NBN = NBN + 1
      DO 14 I=1,N
      DO 14 J=1,N
      G(I,J) = G(I,J) + NBN * (C(I,J) - B(I,J))
   14 B(I,J) = C(I,J)
      IF(NBN .LT. N -1)  GOTO  11
C
C     PRINT HEADING
C
      WRITE(TTYOUT,1)
C
C     COMPUTE PATHS
C
      NCPT = 0
      DO 16 IGP=2,N
      LIM = IGP - 1
      DO 16 JGP=1,LIM
      NB = G(IGP,JGP)
      NA = G(JGP,IGP)
      IF(NA .EQ. 0 .OR. NB .EQ. 0)  GOTO  16
      NBND = 1
      LIDN(NBND) = IGP
      IF(NB .GT. 1)  GOTO  17
C
      NBND = NBND + 1
      LIDN(NBND) = JGP
      GOTO  21
C
   19 WRITE(TTYOUT,2)  NB,IGP,JGP
      GOTO  21
   17 LMM = NB - 1
      LAST = 0
      DO 20 IN=1,LMM
      NBND = NBND + 1
      LIDN(NBND) = NOTR(G,IGP,NB-IN,JGP,IN,IX,N,LAST)
      LAST = LIDN(NBND)
   20 IF(IX .EQ. 1)  GOTO  19
C
      NBND = NBND + 1
      LIDN(NBND) = JGP
   21 IF(NA .GT. 1)  GOTO  22
      LIDN(NBND+1) = IGP
      GOTO  23
C
   24 WRITE(TTYOUT,2)  NA,IGP,JGP
      GOTO  23
C
   22 LMN = NA - 1
      LAST = 0
      DO 25 IN=1,LMN
      NBND = NBND + 1
      LIDN(NBND) = NOTR(G,JGP,NA-IN,IGP,IN,IX,N,LAST)
      LAST = LIDN(NBND)
   25 IF(IX .EQ. 1)  GOTO  24
C
      LIDN(NBND+1) = IGP
   23 CONTINUE
      NCPT = NCPT + 1
      LMC = NBND + 1
C
C     FIX PRINT OUT
C
      DO 15 NX = 1,LMC
      NUM = LIDN(NX)
   15 PRINT(NX) = INDEX(NUM)
      NXXX = NA + NB
      WRITE(TTYOUT,3) NCPT,NXXX,INDEX(IGP),INDEX(JGP),(PRINT(I),I=1,LMC)
   16 CONTINUE
      RETURN
C
C     FORMATS
C
    1 FORMAT('-','NUMBER',5X,'#LINKS',4X,'ELEMENTS',5X,'PATH')
    2 FORMAT(1X,' PAS TROUVE ',3I5)
    3 FORMAT(' ',1X,I4,8X,I2,3X,I5,'-',I5,3X,6(7(I5),/33X))
      END
      INTEGER FUNCTION NOTR(G,ICL,NBLG,ILG,NBCL,IX,NAN,LAST)
C
      IMPLICIT   INTEGER  (A-Z)
      INTEGER  G(50,50),  LST(50)
C
      IX = 0
      IBI = 0
      DO 1 I=1,NAN
      IF(G(I,ILG) .NE. NBLG)  GOTO  1
      IBI = IBI + 1
      LST(IBI) = I
    1 CONTINUE
      IF(IBI .LE. 0)  RETURN
      DO 2 I=1,NBN
      IF(G(ICL,I) .NE. NBCL)  GOTO  2
      IF(LAST .EQ. 0)  GOTO  5
      IF(G(LAST,I) .NE. 1)  GOTO  2
    5 DO 3 K=1,IBI
    3 IF(I .EQ. LST(K))  GOTO  4
    2 CONTINUE
      IX = 1
      RETURN
    4 NOTR = LST(K)
      RETURN
      END
      PROGRAM MAKEIT
C
C     ****************************************************************
C     *                           NOTICE                             *
C     *                                                              *
C     *  ALL RIGHTS RESERVED.  NO PART OF THIS PROGRAM MAY BE SOLD,  *
C     *  REPRODUCED, STORED IN A RETRIEVAL SYSTEM, OR TRANSMITTED    *
C     *  IN ANY FORM OR BY ANY MEANS, ELECTRONIC, MECHANICAL,        *
C     *  PHOTOCOPYING, RECORDING, OR OTHERWISE, WITHOUT THE          *
C     *  PRIOR PERMISSION OF THE                                     *
C     *       UNIVERSITY OF DAYTON RESEARCH INSTITUTE.               *
C     *                                                              *
C     ****************************************************************
C
C
C     PROGRAM AUTHOR: DAVID R. YINGLING, JR.
C               DATE: DECEMBER 31, 1976
C
C     THIS PROGRAM CONVERTS A SEQUENTIAL ACCESSED EDITOR
C     CREATED FILE INTO A DIRECT ACCESS FILE FOR
C     FULL TEXT QUERIES
C
      IMPLICIT INTEGER (A-Z)
C
C     THE ENGLISH TEXT CAN BE 60 CHARS. ON ONE LINE
C     AND UP TO TEN LINES
C
      COMMON  TTYIN, TTYOUT, N, TOTAL, POS, K,  LENGTH(10),  PACKED(200)
      DATA YUP/1HY/
C
C     SYSTEM DEPENDENT INITIALIZATIONS
C
C     THE DIMENSION OF ARRAY "PACKED" MUST BE
C     CHANGED TO "NWORDS"
C
C     EACH EBCDIC OR BCD LETTER USES X BITS
C
C        FOR IBM 360/370        X=8
C        FOR CDC 6600/6400      X=6
C
C     NWORDS IS EQU TO 60 CHARACTERS (ONE LINE) DIVIDED BY
C     NUMBER OF CHARACTERS ABLE TO BE STORED IN ONE WORD (NCHAR) TIMES
C     TEN (FOR TEN LINES).
C
C     NCHAR = NUMBER OF BITS PER MACHINE WORD (NBITS) DIVIDED
C     BY "X".
C
      DIMENSION   CARD(60)
C
      TTYIN  = 1
      TTYOUT = 2
      NBITS  = 32
      X      = 8
      NCHAR  = NBITS / X
      NWORDS = (60 / NCHAR) * 10
      TOTAL  = NWORDS + 10
C
C     DEFINE DIRECT ACCESS FILE
```

```
C     FORTRAN UNIT # = 8
C     RECSIZE (FOR 32 BITS) = 160
C     NUM OF RECORDS =260
C
C
      DEFINE FILE 8(260,160,U,UNUSED)
C
C     CONSTRUCT FILE
C
      CALL DOIT(NCHAR,NWORDS,CARD)
C
C     N IS TH NUMBER OF TEXT RECORDS AND FIRST RECORD OF FILE
C
      WRITE(8'1)  N
C
C     SEE IF USER WANTS TO DISPLAY ELEMENTS
C
      WRITE(TTYOUT,1)
      READ (TTYIN,2)  REPLY
      IF(REPLY .EQ. YUP)  CALL SHOW
      CALL EXIT
C
C     SATISFY THE COMPILER WITH STOP STATEMENT
C
      STOP11111
C
C     FORMATS
C
    1 FORMAT(1H0,47HPERMFILE HAS BEEN CREATED FOR FULL TEXT QUERIES/1H ,
     +11HSHOW(Y/N) ?)
    2 FORMAT(A1)
      END
      SUBROUTINE QUIT(NCHAR,NWORDS,CARD)
C
C     THIS SUBROUTINE CREATES THE RANDOM TEXT FILE
C
      IMPLICIT INTEGER (A-Z)
      COMMON  TTYIN,TTYOUT,N,TOTAL,POS,START,LENGTH(10),PACKED(200)
      LOGICAL  FLAG
      DATA  R/1HR/, E/1HE/, L/1HL/, SLANT/1H//, ONE/1H1/, TWO/1H2/
      DATA  THREE/1H3/
      DIMENSION CARD(60)
C
      START = 1
      POS    = 0
      FLAG   = .FALSE.
      N      = 0
C
C     ZAP OUT LENGTH INDICATORS
C
      DO 1 I=1,10
      LENGTH(I) = 0
    1 CONTINUE
C
C     FORTRAN UNIT 10 POINTS TO SEQUENTIAL TEXT FILE
C
    2 READ(10,10,END=3)  CARD
      IF(CARD(1) .EQ. SLANT)  GOTO  4
C
C     NOT A CONTROL KEYWORD, PACK MORE TEXT INTO "PACKED"
C
      CALL PACK(NWORDS,NCHAR,CARD)
      GOTO  2
C
C     SEE IF FLAG IS ON, IF SO, WRITE CURRENT "PACKED" AND PROCESS
C     SLANT RECORD
C
    4 IF(.NOT. FLAG)  GOTO  5
C
C     WRITE RECORD TO FILE ACCORDING TO LOGPOS (LOGICAL POSITION)
C
      CALL WRITE(LOGPOS,NWORDS)
C
C     PROCESS SLANT RECORD
C
    5 LOGPOS = 0
      IF(CARD(2) .NE. R)  GOTO  6
C
C
C
C
C
C
C
C
C     THIS IS A RELATIONAL CLAUSE, WHICH ONE ?
C
      IF(CARD(3) .EQ. ONE)    LOGPOS = 2
      IF(CARD(3) .EQ. TWO)    LOGPOS = 3
      IF(CARD(3) .EQ. THREE)  LOGPOS = 4
      IF(LOGPOS .LT. 2)  GOTO  9
      FLAG = .TRUE.
      GOTO  2
C
C     COULD BE A "/EL" CARD
C
    6 IF(CARD(2) .NE. E)  GOTO  8
      IF(CARD(3) .EQ. L)  LOGPOS = N + 5
      IF(LOGPOS .LT. 2)  GOTO  9
      FLAG = .TRUE.
      GOTO  2
C
C     IT SHOULD BE AN EOF "//" CARD
C
    8 IF(CARD(2) .EQ. SLANT)  RETURN
C
C     ITS NOT ANYTHING RECOGNIZABLE
C
      GOTO  9
    3 WRITE(TTYOUT,11)
    9 WRITE(TTYOUT,12)
      CALL EXIT
C
C     ISSUE STOP STATEMENT TO SATISFY COMPILER
C
      STOP22222
C
C     FORMATS
C
   10 FORMAT(64A1)
   11 FORMAT(1H0,43H***MISSING "//" CARD AT END OF TEXT FILE***)
   12 FORMAT(1H0,45H***ERROR*** INVALID SLANT KEYWORD ENCOUNTERED/1H ,28
     *HRE-EDIT TEXT FILE TO CORRECT/1H-,23H***MAKEIT TERMINATED***)
      END
      SUBROUTINE WRITE(LOGPOS,NWORDS)
C
C     THIS SUBROUTINE WRITES OUT "PACKED" AND "LENGTH" ONTO
C     THE DIRECT ACCESS FILE
C
      IMPLICIT INTEGER (A-Z)
      COMMON  TTYIN,TTYOUT,N,TOTAL,POS,START,LENGTH(10),PACKED(200)
      IF(LOGPOS .LT. 5)  GOTO  1
C
C     THIS IS AN ELEMENT'S TEXT
C
      N = N + 1
C
C     WRITE ONTO FILE
C
    1 WRITE(8'LOGPOS)  (LENGTH(I), I=1,TOTAL)
      START = 1
      POS    = 0
C
C     ZAP OUT LENGTH INDICATOR
C
      DO 2 I=1,10
      LENGTH(I) = 0
    2 CONTINUE
      RETURN
      END
      SUBROUTINE PACK(NWORDS,NCHAR,CARD)
C
C     THIS SUBROUTINE PACKS:
C
C        1) AS MANY CHARACTERS AS POSSIBLE INTO ONE MACHINE WORD
C
C        2) THE LINES OF TEXT ON A WORD BOUNDRY INTO "PACKED"
C
C
      IMPLICIT INTEGER (A-Z)
      COMMON  TTYIN,TTYOUT,N,TOTAL,POS,START,LENGTH(10),PACKED(200)
      DIMENSION  CARD(60)
      DATA BLANK/1H /
C
C     FIGURE OUT LENGTH
C
      POS = POS + 1
      DO 1 I=1,60
      IF(CARD(61 - I) .NE. BLANK)  GOTO  2
    1 CONTINUE
C
      I   = 59
    2 LEN = 61 - I
C
C     NOW COMES THE TRICKY PART.  RE-READ INFORMATION TO PACK.
C     WITH CDC EXTENDED FORTRAN USE AN ENCODE STATEMENT
C     ON OTHER COMPUTERS USE CORE TO CORE I/O OR A SCRATCH
C     I/O UNIT.  FORTRAN UNIT 20 FOR MY SPECTRA 70 IS VIRTUAL MEMORY
C
      CORE = 20
      REWIND CORE
      WRITE(CORE,3)  (CARD(I),I=1,LEN)
      REWIND CORE
C
C     FIGURE OUT LENGTH OF PACKED LINE
C
      LENGTH(POS) = (LEN - 1) / NCHAR + 1
      END = START + LENGTH(POS)
      READ(CORE,4)  (PACKED(I),I=START,END)
      START = START + LENGTH(POS)
      RETURN
C
C     BE SURE TO CHECK FORMAT # 4
C
    3 FORMAT(60A1)
    4 FORMAT(15A4)
      END
      SUBROUTINE SHOW
C
C     THIS SUBROUTINE SHOWS THE ELEMENTS AS THEY MIGHT
C     APPEAR DURING AN ISM SESSION. THIS IS HELPFUL
C     FOR DETERMINING THE READABILITY OF THE TEXT
C
      IMPLICIT INTEGER (A-Z)
      COMMON  TTYIN, TTYOUT, N, TOTAL, POS, START, LENG(10),DUM(200)
      DATA YUP/1HY/
      COMMON /SHOWB/ R1(160), L1(160), R2(160), L2(160), R3(160)
C
C     COMMON "SHOWB" AND VECTOR "BLOCK" ARE EQU AND WILL
C     CONTAIN THE TEXT
C
      LOGICAL ALLS
      DIMENSION BLOCK(850), NUMS(2)
      EQUIVALENCE (BLOCK,R1), (EL1,NUMS(1)), (EL2,NUMS(2))
      FIND (8'2)
      ALLS = .FALSE.
      EL1  = 0
      EL2  = 0
C
C     READ IN RELATIONAL CLAUSES 1,2, + 3
C
      READ(8'2)  (R1(I),I=1,TOTAL)
      READ(8'3)  (R2(I),I=1,TOTAL)
      READ(8'4)  (R3(I),I=1,TOTAL)
C
C     SEE IF USER WANTS TO DISPLAY ALL ELEMENTS
C
      WRITE(TTYOUT,9)
      READ (TTYIN,10)  REPLY
      IF(REPLY .EQ. YUP)  GOTO  2
C
C     SEE WHICH ELEMENTS THE USER WANTS TO SHOW
C
    1 WRITE(TTYOUT,5)
      CALL GETNUM(NUMS,2)
      IF(EL1 .EQ. 0 .OR. EL2 .EQ. 0)  RETURN
      IF(EL1 .LE. N .AND. EL2 .LE. N)  GOTO  8
      IF(EL1 .GT. N)  WRITE(TTYOUT,11)  EL1
      IF(EL2 .GT. N)  WRITE(TTYOUT,11)  EL2
      GOTO  1
    8 I1 = EL1 + 4
      FIND (8'I1)
      I2 = EL2 + 4
      READ(8'I1)  (L1(I),I=1,TOTAL)
      READ(8'I2)  (L2(I),I=1,TOTAL)
      OFFSET = 0
      DO 3 I=1,5
      I1 = 0
      I2 = 0
      DO 4 J=1,10
      IF(BLOCK(J + OFFSET) .EQ. 0)  GOTO  4
      LENGTH = BLOCK(J + OFFSET)
      I1 = I2 + 1
      I2 = I1 + LENGTH - 1
      WRITE(TTYOUT,7)  (BLOCK(C + OFFSET + 10),C=I1,I2)
    4 CONTINUE
      OFFSET = OFFSET + TOTAL
    3 CONTINUE
      IF(.NOT. ALLS)  GOTO  1
C
    2 EL1 = EL2 + 1
      EL2 = EL1 + 1
      IF(EL1 .GT. N)  RETURN
      IF(EL2 .GT. N)  EL2 = 1
      ALLS = .TRUE.
      GOTO  8
C
C
C     FORMATS
C
    5 FORMAT(1H ,25HSHOW WHICH TWO ELEMENTS ?)
    6 FORMAT(2I5)
    7 FORMAT(1H ,15A4)
    9 FORMAT(1H-,24HSHOW ALL ELEMENTS(Y/N) ?)
   10 FORMAT(A1)
   11 FORMAT(1H-,11H***ERROR***,I5,24H NOT FOUND ON QUERY FILE)
      END
      SUBROUTINE GETNUM(ARRAY,N)
C
C     THIS SUBROUTINE WILL READ "N" UNSIGNED INTEGERS FROM
C     THE TERMINAL TYPED IN A FREE FORMAT AND STORE THEM IN
C     "ARRAY"
C
C     WRITTEN BY:  DAVID R. YINGLING, JR.
C                  DECISION SYSTEMS LAB
C                  ENGINEERING AND PUBLIC POLICY GROUP
C                  UNIVERSITY OF DAYTON
C                  DAYTON, OHIO  45469
C                  513-229-2238
C
      IMPLICIT INTEGER (A-Z)
      COMMON TTYIN, TTYOUT, N, TOTAL, POS, START, LENG(10), DUM(200)
      DIMENSION ARRAY(1), BUFFER(80), NUMS(10)
      DATA NUMS/1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9/
      DATA BLANK/1H /, COMMA/1H,/
C
      DO 8 I=1,N
      ARRAY(I) = 0
    8 CONTINUE
C
    1 READ(TTYIN,200)  BUFFER
C
      L          = N
      POWER      = 0
C
      DO 2 I=1,80
      K = 81 - I
      IF(BUFFER(K) .EQ. BLANK .OR. BUFFER(K) .EQ. COMMA)  GOTO  3
C
C     FOUND A CHARACTER, SEE IF IT'S A VALID NUMERIC
C
      DO 4 J=1,10
      II = J - 1
      IF(BUFFER(K) .NE. NUMS(J))  GOTO  4
C
C     ITS A NUMBER, ADD IT TO PRESENT SUM
C
      ARRAY(L) = ARRAY(L) + (II * (10**POWER))
      POWER      = POWER + 1
      GOTO  2
    4 CONTINUE
C
C     COME HERE IF CHARACTER FOUND WAS NOT NUMERIC
C
      WRITE(TTYOUT,100)
      GOTO  1
C
C     FOUND A DELIMITER, SEE IF END OF A NUMBER
C
    3 IF(ARRAY(L) .EQ. 0)  GOTO  2
      L = L - 1
      IF(L .EQ. 0)  GOTO  5
      ARRAY(L) = 0
      POWER      = 0
    2 CONTINUE
```

```
C
C     MAKE SURE NUMBER(S) IS/ARE LESS THAN 99999 SO
C     WE DON'T EXCEED I5 FORMATS
C
    5 DO 6 I=1,N
      IF(ARRAY(I) .GT. 99999)  GOTO  7
    6 CONTINUE
      RETURN
C
C     ERROR MESSAGE
C
    7 WRITE(TTYOUT,101)
      GOTO  1
C
C     FORMATS
C
  100 FORMAT(37H-***ERROR*** INPUT NOT NUMERIC--RETRY)
  101 FORMAT(39H-***ERROR*** NUMBER(S) TOO LARGE--RETRY)
  200 FORMAT(80A1)
      END
```

Distribution

Copy No.

1 - 6 Mr. Walter Bogan, Director
Office of Environmental Education
400 Maryland Ave. S. W.
FOB #6, Room 2025
Washington, D. C. 20202

7 - 8 Mr. George Coates
Office of Environmental Education
400 Maryland Avenue S. W.
FOB #6, Room 2025
Washington, D. C. 20202

9 Dr. Alexander Christakis
University of Virginia
Engineering Science and Systems Department
Room 234 A & M Building

10 Bro. Raymond Fitz
Office of the President
University of Dayton
Dayton, Ohio 45469

11 Dr. H. Grant Goodell
Department of Environmental Sciences
Clark Hall
University of Virginia

12 Dr. R. W. House
Box 6188, Station B
School of Engineering
Vanderbilt University
Nashville, Tennessee 37235

13 Dr. Robert Waller
Department of Business
University of Northern Iowa
Cedar Falls, Iowa 50613

14 Office of Sponsored Programs
Madison Hall
University of Virginia

15 - 24 J. N. Warfield
Department of Electrical Engineering
University of Virginia

Distribution (cont.)

25 - 26 MS E. H. Pancake
Science/Technology Information Center
Clark Hall
University of Virginia

27 RLES files

28 Professor Robert Stake
CIRCE
College of Education
University of Illinois
Urbana, IL 61801

29 Dr. Tom Hastings
CIRCE
College of Education
University of Illinois
Urbana, IL 61801

30 Dr. Bela Banathy
Far West Laboratory
1855 Folsom Street
San Francisco, CA 94103

UNIVERSITY OF VIRGINIA

School of Engineering and Applied Science

The University of Virginia's School of Engineering and Applied Science has an undergraduate enrollment of approximately 1,300 students with a graduate enrollment of approximately 500. There are 125 faculty members, a majority of whom conduct research in addition to teaching.

Research is an integral part of the educational program and interests parallel academic specialties. These range from the classical engineering departments of Chemical, Civil, Electrical, and Mechanical and Aerospace to departments of Biomedical Engineering, Engineering Science and Systems, Materials Science, Nuclear Engineering and Engineering Physics, and Applied Mathematics and Computer Science. In addition to these departments, there are interdepartmental groups in the areas of Automatic Controls and Applied Mechanics. All departments offer the doctorate; the Biomedical and Materials Science Departments grant only graduate degrees.

The School of Engineering and Applied Science is an integral part of the University (approximately 1,530 full-time faculty with a total enrollment of about 16,000 full-time students), which also has professional schools of Architecture, Law, Medicine, Commerce, and Business Administration. In addition, the College of Arts and Sciences houses departments of Mathematics, Physics, Chemistry and others relevant to the engineering research program. This University community provides opportunities for interdisciplinary work in pursuit of the basic goals of education, research, and public service.